Weighted Automata

with Ambiguity and Extensions

Ritam Raha [1] Nathanaël Fijalkow [2] Filip Mazowiecki [2]
Vincent Penelle [2] Nathan Lhote [2]

[1]Chennai Mathematical Institute

[2]LaBRI, Bordeaux

Formal Methods and Verification Seminar - ULB
December 4, 2018

Outline

1. Weighted Automata
2. Hankel Matrix
3. Ambiguity
4. Universality with Ambiguity
5. Introduction to Weighted Context-Free Grammar
6. Learning WCFG
7. Properties of WCFG

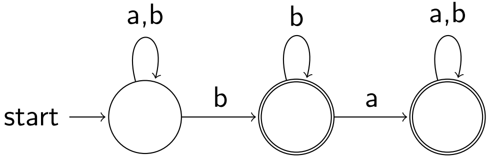
Outline

1. Weighted Automata
2. Hankel Matrix
3. Ambiguity
4. Universality with Ambiguity
5. Introduction to Weighted Context-Free Grammar
6. Learning WCFG
7. Properties of WCFG

Weighted Automata:

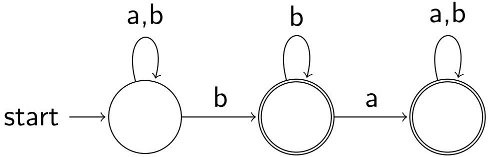
Weighted Automata:

Automata



Weighted Automata:

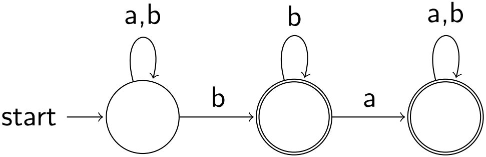
Automata



$f : \Sigma^* \rightarrow \{0, 1\}$

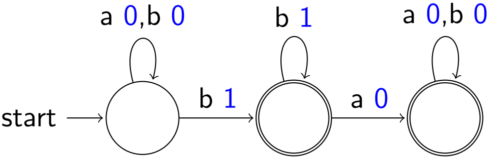
Weighted Automata:

Automata



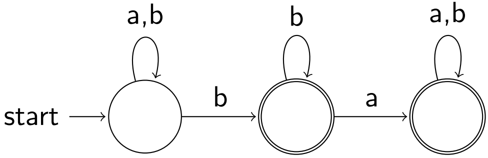
$f : \Sigma^* \to \{0, 1\}$

Weighted Automata



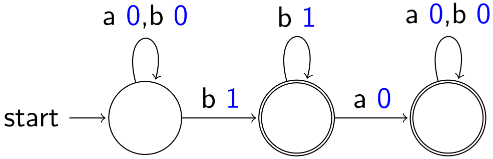
Weighted Automata:

Automata



$f : \Sigma^* \to \{0, 1\}$

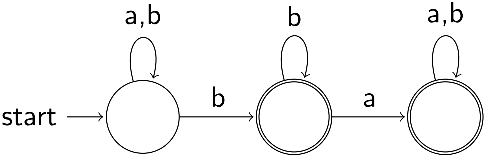
Weighted Automata



$f : \Sigma^* \to S$

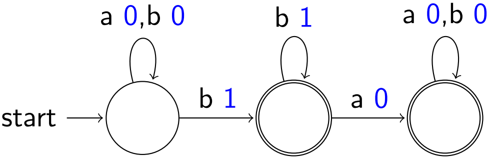
Weighted Automata:

Automata



$f : \Sigma^* \rightarrow \{0, 1\}$

Weighted Automata



$f : \Sigma^* \rightarrow S$ **(Semiring)**

Weighted Automata:

Semiring

$S(\oplus, \odot, 0, \mathbb{1})$

Weighted Automata:

Semiring

$S(\oplus, \odot, \mathbb{0}, \mathbb{1})$

Examples:

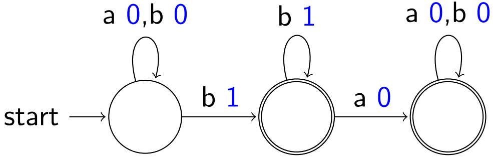
- Natural Semiring : $\mathbb{N}(+, \cdot, 0, 1)$
- Tropical Semiring:

$$\mathbb{N}_{\infty}(\text{min}, +, \infty, 0) \text{ Min-plus Semiring}$$

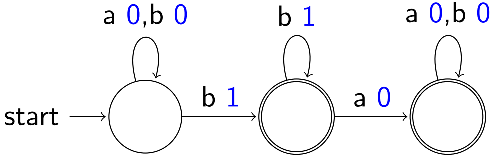
or

$$\mathbb{N}_{-\infty}(\text{max}, +, -\infty, 0) \text{ Max-plus Semiring}$$

Weighted Automata:

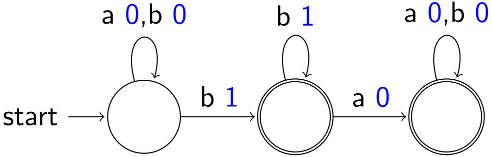


Weighted Automata:



Max-plus Semiring

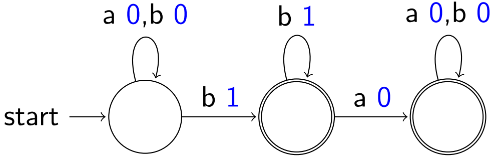
Weighted Automata:



Max-plus Semiring

Consider the word *bbab*:

Weighted Automata:

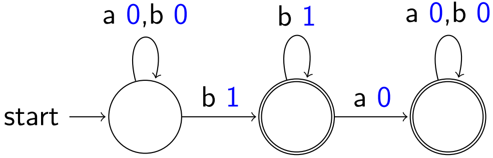


Max-plus Semiring

Consider the word *bbab*:

b b a b	b b a b	b b a b
1 +1 +0 +0=2	0 +1 +0 +0=1	0 +0 +0 +1=1

Weighted Automata:



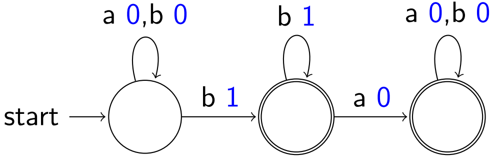
Max-plus Semiring

Consider the word *bbab*:

b b a b	b b a b	b b a b
$1 + 1 + 0 + 0 = 2$	$0 + 1 + 0 + 0 = 1$	$0 + 0 + 0 + 1 = 1$

Output: $\max\{2, 1, 1\} = 2$

Weighted Automata:



Max-plus Semiring

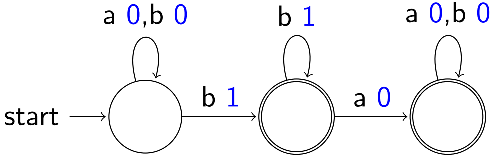
Consider the word *bbab*:

b b a b	b b a b	b b a b
1 +1 +0 +0=2	0 +1 +0 +0=1	0 +0 +0 +1=1

Output: $\max\{2, 1, 1\} = 2$

In general: $\odot$ transitions, $\oplus$ runs

Weighted Automata:



Max-plus Semiring

Consider the word *bbab*:

b b a b	b b a b	b b a b
1 +1 +0 +0=2	0 +1 +0 +0=1	0 +0 +0 +1=1

Output: $\max\{2,1,1\} = 2$
In general: $\odot$ transitions, $\oplus$ runs

Counting the length of the longest *b*-block

Outline

1 Weighted Automata

2 Hankel Matrix

3 Ambiguity

4 Universality with Ambiguity

5 Introduction to Weighted Context-Free Grammar

6 Learning WCFG

7 Properties of WCFG

Hankel Matrix:

Alternatively, we can see a weighted automata $\mathcal{A}$ on a Semiring S like the following:

Hankel Matrix:

Alternatively, we can see a weighted automata $\mathcal{A}$ on a Semiring S like the following:

$\mathcal{A} = \langle Q, \alpha \in S^Q, (\Delta(a) \in S^{Q\times Q})_{a\in\Sigma}, \eta \in S^Q \rangle$

Hankel Matrix:

Alternatively, we can see a weighted automata $\mathcal{A}$ on a Semiring S like the following:

$\mathcal{A} = \langle Q, \alpha \in S^Q, (\Delta(a) \in S^{Q \times Q})_{a \in \Sigma}, \eta \in S^Q \rangle$

$\mathcal{A}$ recognizes a function $f : \Sigma^* \to S$, where

$f(a_1 \ldots a_n) = \alpha . \underbrace{\Delta(a_1) \ldots \Delta(a_n)}_{\Delta(a_1 \ldots a_n)} . \eta$

Hankel Matrix:

Alternatively, we can see a weighted automata $\mathcal{A}$ on a Semiring S like the following:
$\mathcal{A} = \langle Q, \alpha \in S^Q, (\Delta(a) \in S^{Q \times Q})_{a \in \Sigma}, \eta \in S^Q \rangle$
$\mathcal{A}$ recognizes a function $f : \Sigma^* \to S$, where
$f(a_1 \dots a_n) = \alpha . \underbrace{\Delta(a_1) \dots \Delta(a_n)}_{\Delta(a_1 \dots a_n)} . \eta$

Now, consider a bi-infinite matrix $H_f \in S^{\Sigma^* \times \Sigma^*}$, such that $H_f(u, v) = f(uv)$.

Hankel Matrix:

Alternatively, we can see a weighted automata $\mathcal{A}$ on a Semiring S like the following:

$\mathcal{A} = \langle Q, \alpha \in S^{Q}, (\Delta(a) \in S^{Q \times Q})_{a \in \Sigma}, \eta \in S^{Q} \rangle$

$\mathcal{A}$ recognizes a function $f : \Sigma^{*} \to S$, where

$$f(a_1 \ldots a_n) = \alpha . \underbrace{\Delta(a_1) \ldots \Delta(a_n)}_{\Delta(a_1 \ldots a_n)} . \eta$$

Now, consider a bi-infinite matrix $H_f \in S^{\Sigma^{*} \times \Sigma^{*}}$, such that $H_f(u, v) = f(uv)$.

$$H_f = \begin{array}{cc} & \begin{array}{ccc} & v & \end{array} \\ u & \left(\begin{array}{cccc} & & . & \\ & & . & \\ . & . & f(uv) & \\ & & & \end{array}\right) \end{array}$$

Hankel Matrix:

Alternatively, we can see a weighted automata $\mathcal{A}$ on a Semiring S like the following:
$\mathcal{A} = \langle Q, \alpha \in S^Q, (\Delta(a) \in S^{Q \times Q})_{a \in \Sigma}, \eta \in S^Q \rangle$
$\mathcal{A}$ recognizes a function $f : \Sigma^* \to S$, where
$f(a_1 \dots a_n) = \alpha . \underbrace{\Delta(a_1) \dots \Delta(a_n)}_{\Delta(a_1 \dots a_n)} . \eta$

Now, consider a bi-infinite matrix $H_f \in S^{\Sigma^* \times \Sigma^*}$, such that $H_f(u, v) = f(uv)$.

$$H_f = \begin{array}{cc} & \begin{array}{cccc} & & v & \end{array} \\ u & \left(\begin{array}{cccc} & & . & \\ & & . & \\ . & . & f(uv) & \\ & & & \end{array}\right) \end{array}$$

This is called **Hankel Matrix**.

Hankel Matrix:

Theorem: (Fliess '74) [Fijb]

- Any automaton recognizing f has at least rank(H_f) many states,
- There exists an automaton recognizing f with rank(H_f) many states.

Hankel Matrix:

Theorem: (Fliess '74) [Fijb]

- Any automaton recognizing f has at least $\mathrm{rank}(H_f)$ many states,
- There exists an automaton recognizing f with $\mathrm{rank}(H_f)$ many states.

Application:

- Given a rational function f, we can effectively construct the minimal weighted automaton recognizing f.

Hankel Matrix:

Theorem: (Fliess '74) [Fijb]

- Any automaton recognizing f has at least $\text{rank}(H_f)$ many states,
- There exists an automaton recognizing f with $\text{rank}(H_f)$ many states.

Application:

- Given a rational function f, we can effectively construct the minimal weighted automaton recognizing f.
- Weighted automata over the reals can be learned efficiently in Angluin's supervised scenario. [Fija]

Hankel Matrix:

Theorem: (Fliess '74) [Fijb]

- Any automaton recognizing f has at least $\mathrm{rank}(H_f)$ many states,
- There exists an automaton recognizing f with $\mathrm{rank}(H_f)$ many states.

Application:

- Given a rational function f, we can effectively construct the minimal weighted automaton recognizing f.
- Weighted automata over the reals can be learned efficiently in Angluin's supervised scenario. [Fija]
- Some more applications will follow...

Outline

1. Weighted Automata
2. Hankel Matrix
3. Ambiguity
4. Universality with Ambiguity
5. Introduction to Weighted Context-Free Grammar
6. Learning WCFG
7. Properties of WCFG

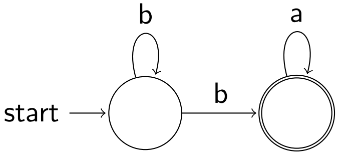
Ambiguity:

Counts the number of accepting runs of a word!

Ambiguity:

Counts the number of accepting runs of a word!

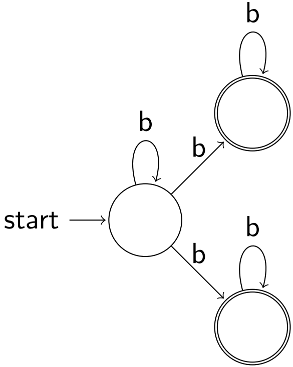
If all words have maximum one accepting run - Unambiguous



Ambiguity:

Counts the number of accepting runs of a word!

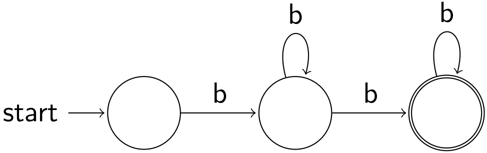
If all words have finitely many accepting run - Finite ambiguous



Ambiguity:

Counts the number of accepting runs of a word!

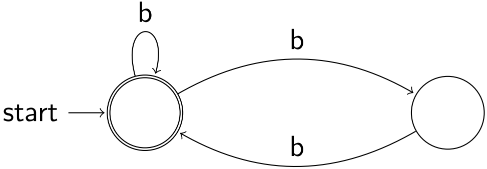
If the maximum degree of ambiguity is bounded by some polynomial in the length of the word - Polynomially ambiguous



Ambiguity:

Counts the number of accepting runs of a word!

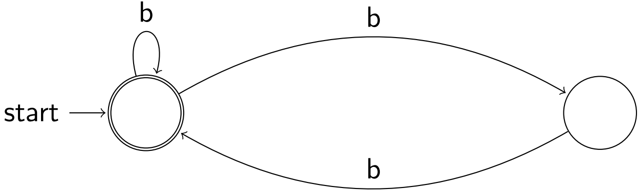
If the degree of ambiguity is not bounded- Exponentially ambiguous



Ambiguity:

Counts the number of accepting runs of a word!

If the degree of ambiguity is not bounded- Exponentially ambiguous



It can be shown that these are the only options for ambiguity of an automata.

Outline

1. Weighted Automata
2. Hankel Matrix
3. Ambiguity
4. Universality with Ambiguity
5. Introduction to Weighted Context-Free Grammar
6. Learning WCFG
7. Properties of WCFG

Universality Problem:

Given an automaton M on alphabet Σ, is $L(M) = \Sigma^*$?

Universality Problem:

Given an automaton M on alphabet Σ, is $L(M) = \Sigma^*$?

Universality problem for any general NFA is PSPACE-complete.

Universality Problem:

Given an automaton M on alphabet Σ, is $L(M) = \Sigma^*$?

Universality problem for any general NFA is PSPACE-complete.

What happens with ambiguity?

Universality Problem:

Lemma:

If there exists a word w that is not accepted by an unambiguous NFA M, then there exists a word w' such that $|w'| \leq |M|$ and w' is not accepted by M.

Universality Problem:

Lemma:

If there exists a word w that is not accepted by an unambiguous NFA M, then there exists a word w' such that $|w'| \leq |M|$ and w' is not accepted by M.

Proof Idea:

Universality Problem:

Lemma:

If there exists a word w that is not accepted by an unambiguous NFA M, then there exists a word w' such that $|w'| \leq |M|$ and w' is not accepted by M.

Proof Idea: Let the shortest word be $u = x_1 x_2 \ldots x_n$, where $n > |M|$.

Universality Problem:

Lemma:

If there exists a word w that is not accepted by an unambiguous NFA M, then there exists a word w' such that $|w'| \leq |M|$ and w' is not accepted by M.

Proof Idea: Let the shortest word be $u = x_1 x_2 \ldots x_n$, where $n > |M|$.

$$H = \begin{pmatrix} \\ \\ \\ \end{pmatrix}$$

Universality Problem:

Lemma:

If there exists a word w that is not accepted by an unambiguous NFA M, then there exists a word w' such that $|w'| \leq |M|$ and w' is not accepted by M.

Proof Idea: Let the shortest word be $u = x_1 x_2 \dots x_n$, where $n > |M|$.

$$H = \begin{array}{c} \begin{matrix} u & . & . & \dots & \epsilon \end{matrix} \\ \left(\begin{matrix} \\ \\ \\ \\ \end{matrix} \right) \end{array}$$

Universality Problem:

Lemma:

If there exists a word w that is not accepted by an unambiguous NFA M, then there exists a word w' such that $|w'| \leq |M|$ and w' is not accepted by M.

Proof Idea: Let the shortest word be $u = x_1 x_2 \dots x_n$, where $n > |M|$.

$$H = \begin{array}{cccccc} & u & . & . & \dots & \epsilon \\ \epsilon & & & & & \\ x_1 & & & & & \\ x_1 x_2 & & & & & \\ \vdots & & & & & \\ u & & & & & \end{array}$$

Universality Problem:

Lemma:

If there exists a word w that is not accepted by an unambiguous NFA M, then there exists a word w' such that $|w'| \leq |M|$ and w' is not accepted by M.

Proof Idea: Let the shortest word be $u = x_1 x_2 \dots x_n$, where $n > |M|$.

$$H = \begin{array}{c|ccccc} & u & . & . & \dots & \epsilon \\ \hline \epsilon & 0 & & & & \\ x_1 & & 0 & & & \\ x_1 x_2 & & & 0 & & \\ \vdots & & & & 0 & \\ u & & & & & 0 \end{array}$$

Universality Problem:

Lemma:

If there exists a word w that is not accepted by an unambiguous NFA M, then there exists a word w' such that $|w'| \leq |M|$ and w' is not accepted by M.

Proof Idea: Let the shortest word be $u = x_1 x_2 \ldots x_n$, where $n > |M|$.

$$H = \begin{array}{c} \\ \epsilon \\ x_1 \\ x_1x_2 \\ \vdots \\ u \end{array} \begin{array}{c} \begin{array}{ccccc} u & . & . & \ldots & \epsilon \end{array} \\ \begin{pmatrix} 0 & & & & \\ & 0 & & \mathbf{1} & \\ & & 0 & & \\ & & & 0 & \\ & & & & 0 \end{pmatrix} \end{array}$$

Universality Problem:

Lemma:

If there exists a word w that is not accepted by an unambiguous NFA M, then there exists a word w' such that $|w'| \leq |M|$ and w' is not accepted by M.

Proof Idea: Let the shortest word be $u = x_1 x_2 \dots x_n$, where $n > |M|$.

$$H = \begin{array}{cccccc} & u & . & . & \dots & \epsilon \\ \epsilon & 0 & & & & \\ x_1 & & 0 & & \mathbf{1} & \\ x_1x_2 & & & 0 & & \\ \vdots & & & & 0 & \\ u & & & & & 0 \end{array}$$

$\text{rank}(H) > n > |M|$

Universality Problem:

Consider M weighted automata on $\{\mathbb{R} \cap \{0,1\}, +, \times, 0, 1\}$ with all transition weight 1.

Universality Problem:

Consider M weighted automata on $\{\mathbb{R} \cap \{0,1\}, +, \times, 0, 1\}$ with all transition weight 1.

M computes $f : \Sigma^* \rightarrow \{0,1\}$.

Universality Problem:

Consider M weighted automata on $\{\mathbb{R} \cap \{0,1\}, +, \times, 0, 1\}$ with all transition weight 1.

M computes $f : \Sigma^* \to \{0,1\}$.

M unambiguous

Universality Problem:

Consider M weighted automata on $\{\mathbb{R} \cap \{0,1\}, +, \times, 0, 1\}$ with all transition weight 1.

M computes $f : \Sigma^* \rightarrow \{0,1\}$.

M unambiguous $\Rightarrow H \subset H_f$

Universality Problem:

Consider M weighted automata on $\{\mathbb{R} \cap \{0,1\}, +, \times, 0, 1\}$ with all transition weight 1.

M computes $f : \Sigma^* \to \{0,1\}$.

M unambiguous $\Rightarrow H \subset H_f \Rightarrow rank(H) < |M|$.
Contradiction!

Universality Problem:

Consider M weighted automata on $\{\mathbb{R} \cap \{0,1\}, +, \times, 0, 1\}$ with all transition weight 1.

M computes $f : \Sigma^* \to \{0,1\}$.

M unambiguous $\Rightarrow H \subset H_f \Rightarrow rank(H) < |M|$.
Contradiction!

Clearly in co-NP.

Universality Problem:

Consider M weighted automata on $\{\mathbb{R} \cap \{0,1\}, +, \times, 0, 1\}$ with all transition weight 1.

M computes $f : \Sigma^* \to \{0,1\}$.

M unambiguous $\Rightarrow H \subset H_f \Rightarrow rank(H) < |M|$.
Contradiction!

Clearly in co-NP. Can we do better?

Universality Problem:

Linear Recurrence System: Each term of a sequence is a linear function of earlier terms in the sequence.

$$\left\{\begin{array}{l} f(n) = f(n-1) + g(n-1) \\ g(n) = f(n-1) \\ f(0) = 0 \\ g(0) = 1 \end{array}\right\} \Leftrightarrow \left\{\begin{array}{l} f(n) = f(n-1) + f(n-2) \\ f(0) = 0 \\ f(1) = 1 \end{array}\right\}$$

Universality Problem:

Linear Recurrence System: Each term of a sequence is a linear function of earlier terms in the sequence.

$$\left\{ \begin{array}{l} f(n) = f(n-1) + g(n-1) \\ g(n) = f(n-1) \\ f(0) = 0 \\ g(0) = 1 \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{l} f(n) = f(n-1) + f(n-2) \\ f(0) = 0 \\ f(1) = 1 \end{array} \right\}$$

$$\boxed{\text{Fibonacci}}$$

Universality Problem:

An LRS of order k is a sequence $(u_l)_{l\in\mathbb{N}}$ such that,

$$u_l = X \cdot A^l \cdot Y,$$

where, $A \in \mathbb{R}^{k\times k}$ and $X, Y \in \mathbb{R}^k$.

Universality Problem:

An LRS of order k is a sequence $(u_l)_{l \in \mathbb{N}}$ such that,

$$u_l = X \cdot A^l \cdot Y,$$

where, $A \in \mathbb{R}^{k \times k}$ and $X, Y \in \mathbb{R}^k$.

Fibonacci sequence $\Rightarrow F_l = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Universality Problem:

An LRS of order k is a sequence $(u_l)_{l \in \mathbb{N}}$ such that,

$$u_l = X \cdot A^l \cdot Y,$$

where, $A \in \mathbb{R}^{k \times k}$ and $X, Y \in \mathbb{R}^k$.

Fibonacci sequence $\Rightarrow F_l = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

We will use mainly the following two properties of LRS:

Theorem:

- The l-th term of an LRS of order k can be computed in time $O(log(l) \cdot k^3)$.
- Two LRS of order at most k are equal if and only if they agree on the first k terms.

Universality Problem:

$\alpha(l) =$ No. of l- length words accepted by M.
$Acc(l) =$ No. of l-length accepting paths of M.

Universality Problem:

$\alpha(l)$ = No. of l- length words accepted by M.
$Acc(l)$ = No. of l-length accepting paths of M.

Now, clearly $Acc(l) = I.\Delta^l.F$.

Universality Problem:

$\alpha(l) =$ No. of l- length words accepted by M.
$Acc(l) =$ No. of l-length accepting paths of M.

Now, clearly $Acc(l) = I.\Delta^l.F$.
Hence, $(Acc(l))_{l\in\mathbb{N}}$ is an LRS of order n.

Universality Problem:

$\alpha(l)$ = No. of l- length words accepted by M.
$Acc(l)$ = No. of l-length accepting paths of M.

Now, clearly $Acc(l) = I.\Delta^l.F$.
Hence, $(Acc(l))_{l\in\mathbb{N}}$ is an LRS of order n.
Now, $|\Sigma|^l$ is an LRS of order 1.

Universality Problem:

$\alpha(l) =$ No. of l- length words accepted by M.
$Acc(l) =$ No. of l-length accepting paths of M.

Now, clearly $Acc(l) = I.\Delta^l.F$.
Hence, $(Acc(l))_{l \in \mathbb{N}}$ is an LRS of order n.
Now, $|\Sigma|^l$ is an LRS of order 1.

M unambiguous

Universality Problem:

$\alpha(l)$ = No. of l- length words accepted by M.
$Acc(l)$ = No. of l-length accepting paths of M.

Now, clearly $Acc(l) = I.\Delta^l.F$.
Hence, $(Acc(l))_{l\in\mathbb{N}}$ is an LRS of order n.
Now, $|\Sigma|^l$ is an LRS of order 1.

M unambiguous ⇒ Each run corresponds to a word

Universality Problem:

$\alpha(l)$ = No. of l- length words accepted by M.
$Acc(l)$ = No. of l-length accepting paths of M.

Now, clearly $Acc(l) = I.\Delta^l.F$.
Hence, $(Acc(l))_{l\in\mathbb{N}}$ is an LRS of order n.
Now, $|\Sigma|^l$ is an LRS of order 1.

M unambiguous $\Rightarrow$ Each run corresponds to a word $\Rightarrow \alpha = Acc$

Universality Problem:

$\alpha(l) =$ No. of l- length words accepted by M.
$Acc(l) =$ No. of l-length accepting paths of M.

Now, clearly $Acc(l) = I.\Delta^l.F$.
Hence, $(Acc(l))_{l \in \mathbb{N}}$ is an LRS of order n.
Now, $|\Sigma|^l$ is an LRS of order 1.

M unambiguous $\Rightarrow$ Each run corresponds to a word $\Rightarrow \alpha = Acc$
Also enough to check for words up to length n

Universality Problem:

$\alpha(l)$ = No. of l- length words accepted by M.
$Acc(l)$ = No. of l-length accepting paths of M.

Now, clearly $Acc(l) = I.\Delta^l.F$.
Hence, $(Acc(l))_{l\in\mathbb{N}}$ is an LRS of order n.
Now, $|\Sigma|^l$ is an LRS of order 1.

M unambiguous $\Rightarrow$ Each run corresponds to a word $\Rightarrow \alpha = Acc$
Also enough to check for words up to length $n \Rightarrow$ $\boxed{\text{Polynomial Time}}$

Universality Problem:

What happens with finite ambiguity?

Universality Problem:

What happens with finite ambiguity?

Same approach fails!! The number of l-length accepted words does not correspond to l-length accepting paths any more.

Universality Problem:

What happens with finite ambiguity?

Same approach fails!! The number of l-length accepted words does not correspond to l-length accepting paths any more.

Given A, a k- ambiguous automaton(k-fixed). Construct A_p as follows:

Universality Problem:

What happens with finite ambiguity?

Same approach fails!! The number of l-length accepted words does not correspond to l-length accepting paths any more.

Given A, a k- ambiguous automaton(k-fixed). Construct A_p as follows:
Consider a linear order $<$ on states,
States: $Q' = Q \cup Q^2 \cup \cdots \cup Q^p$ separated with at most $(p-1)$ delimiters,
Transitions: if for some state $q \in Q$, $q \xrightarrow{a} q_1$ & $q \xrightarrow{a} q_2 \in \delta$ and $q_1 < q_2$, then $q \xrightarrow{a} (q_1|q_2) \in \delta'$,
Final state: Final states of A_p will be $(\underbrace{q_f|q_f|\cdots|q_f}_{p \text{ times}})$

Universality Problem:

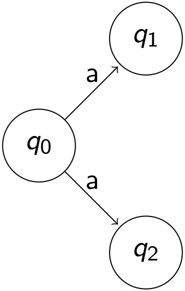
What happens with finite ambiguity?

Same approach fails!! The number of l-length accepted words does not correspond to l-length accepting paths any more.

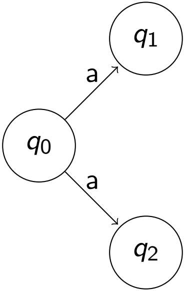
Given A, a k- ambiguous automaton(k-fixed). Construct A_p as follows:
Consider a linear order $<$ on states,
States: $Q' = Q \cup Q^2 \cup \cdots \cup Q^p$ separated with at most $(p-1)$ delimiters,
Transitions: if for some state $q \in Q$, $q \xrightarrow{a} q_1$ & $q \xrightarrow{a} q_2 \in \delta$ and $q_1 < q_2$, then $q \xrightarrow{a} (q_1|q_2) \in \delta'$,
Final state: Final states of A_p will be $(\underbrace{q_f|q_f|\cdots|q_f}_{p \text{ times}})$

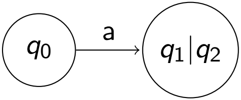
The idea is, we use the powerset construction capped to sets of size at most p with a linear ordering on states.

Universality Problem:

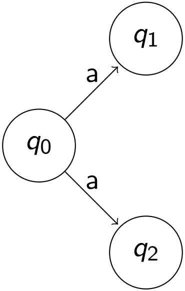


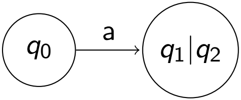
Universality Problem:





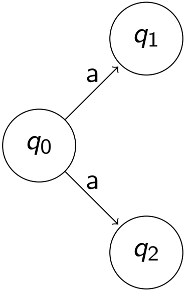
Universality Problem:

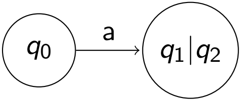




A_p accepts all words that have at least p accepting runs on A.

Universality Problem:





A_p accepts all words that have at least p accepting runs on A.
Also given the linear order on states, A_k is unambiguous, where k is the highest ambiguity.

Universality Problem:

$\alpha(l) =$ the number of words of length l accepted by A,
$\alpha_p(l) =$ the number of words of length l having exactly p accepting runs over A.

Universality Problem:

$\alpha(l) =$ the number of words of length l accepted by A,
$\alpha_p(l) =$ the number of words of length l having exactly p accepting runs over A. $\Rightarrow \alpha(l) = \sum_{p=1}^{k} \alpha_p(l)$

Universality Problem:

$\alpha(l)$ = the number of words of length l accepted by A,
$\alpha_p(l)$ = the number of words of length l having exactly p accepting runs over A. $\Rightarrow \alpha(l) = \sum_{p=1}^{k} \alpha_p(l)$

Note that:

- each word having exactly p runs induce one run of A_p

Universality Problem:

$\alpha(l) =$ the number of words of length l accepted by A,
$\alpha_p(l) =$ the number of words of length l having exactly p accepting runs over A. $\Rightarrow \alpha(l) = \sum_{p=1}^{k} \alpha_p(l)$

Note that:

- each word having exactly p runs induce one run of A_p
- each word having exactly $p+1$ runs induce $p+1$ runs of A_p, obtained by choosing p runs among $p+1$.

Universality Problem:

$\alpha(l)$ = the number of words of length l accepted by A,
$\alpha_p(l)$ = the number of words of length l having exactly p accepting runs over A. $\Rightarrow \alpha(l) = \sum_{p=1}^{k} \alpha_p(l)$

Note that:

- each word having exactly p runs induce one run of A_p
- each word having exactly $p+1$ runs induce $p+1$ runs of A_p, obtained by choosing p runs among $p+1$.
- more generally, each word having exactly j runs induce $\binom{j}{p}$ runs of A_p, obtained by choosing p runs among j.

Universality Problem:

$\alpha(l)$ = the number of words of length l accepted by A,
$\alpha_p(l)$ = the number of words of length l having exactly p accepting runs over A. $\Rightarrow \alpha(l) = \sum_{p=1}^{k} \alpha_p(l)$

Note that:

- each word having exactly p runs induce one run of A_p
- each word having exactly $p+1$ runs induce $p+1$ runs of A_p, obtained by choosing p runs among $p+1$.
- more generally, each word having exactly j runs induce $\binom{j}{p}$ runs of A_p, obtained by choosing p runs among j.

$Acc_p(l)$ = No. of l-length paths in A_p

Universality Problem:

$\alpha(l)$ = the number of words of length l accepted by A,
$\alpha_p(l)$ = the number of words of length l having exactly p accepting runs over A. $\Rightarrow \alpha(l) = \sum_{p=1}^{k} \alpha_p(l)$

Note that:

- each word having exactly p runs induce one run of A_p
- each word having exactly $p+1$ runs induce $p+1$ runs of A_p, obtained by choosing p runs among $p+1$.
- more generally, each word having exactly j runs induce $\binom{j}{p}$ runs of A_p, obtained by choosing p runs among j.

$Acc_p(l)$ = No. of l-length paths in $A_p = \sum_{j=p}^{k} \binom{j}{p} \alpha_j(l)$.

Universality Problem:

Consider $Acc = (Acc_1, Acc_2, \ldots Acc_k)$ and $\alpha = (\alpha_1, \alpha_2, \ldots \alpha_k)$.

Universality Problem:

Consider $Acc = (Acc_1, Acc_2, \ldots Acc_k)$ and $\alpha = (\alpha_1, \alpha_2, \ldots \alpha_k)$.$Acc = M \cdot \alpha$, where M is upper-triangular and invertible.

Universality Problem:

Consider $Acc = (Acc_1, Acc_2, \dots Acc_k)$ and $\alpha = (\alpha_1, \alpha_2, \dots \alpha_k)$.$Acc = M \cdot \alpha$, where M is upper-triangular and invertible.

$\alpha = M^{-1} \cdot Acc$

Universality Problem:

Consider $Acc = (Acc_1, Acc_2, \dots Acc_k)$ and $\alpha = (\alpha_1, \alpha_2, \dots \alpha_k)$.$Acc = M \cdot \alpha$, where M is upper-triangular and invertible.
$\alpha = M^{-1} \cdot Acc$

For each p,
Acc_p is LRS of order $n^{O(k)}$

Universality Problem:

Consider $Acc = (Acc_1, Acc_2, \ldots Acc_k)$ and $\alpha = (\alpha_1, \alpha_2, \ldots \alpha_k)$.$Acc = M \cdot \alpha$, where M is upper-triangular and invertible.
$\alpha = M^{-1} \cdot Acc$

For each p,
Acc_p is LRS of order $n^{O(k)} \Rightarrow \alpha_p$ is LRS of order $n^{O(k)}$

Universality Problem:

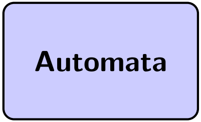
Consider $Acc = (Acc_1, Acc_2, \dots Acc_k)$ and $\alpha = (\alpha_1, \alpha_2, \dots \alpha_k)$.$Acc = M \cdot \alpha$, where M is upper-triangular and invertible.
$\alpha = M^{-1} \cdot Acc$

For each p,
Acc_p is LRS of order $n^{O(k)} \Rightarrow \alpha_p$ is LRS of order $n^{O(k)} \Rightarrow$
$\boxed{\text{Polynomial Time}}$

Outline

1. Weighted Automata
2. Hankel Matrix
3. Ambiguity
4. Universality with Ambiguity
5. Introduction to Weighted Context-Free Grammar
6. Learning WCFG
7. Properties of WCFG

WCFG - nonlinear extension:



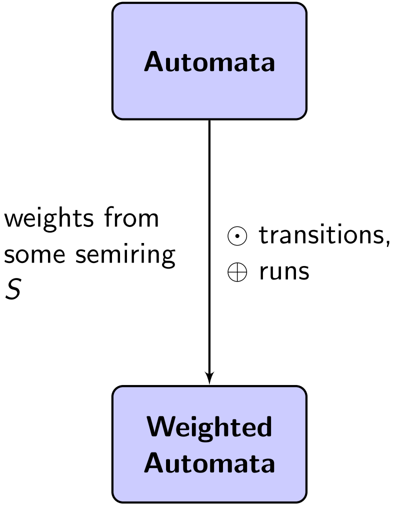
Automata

WCFG - nonlinear extension:

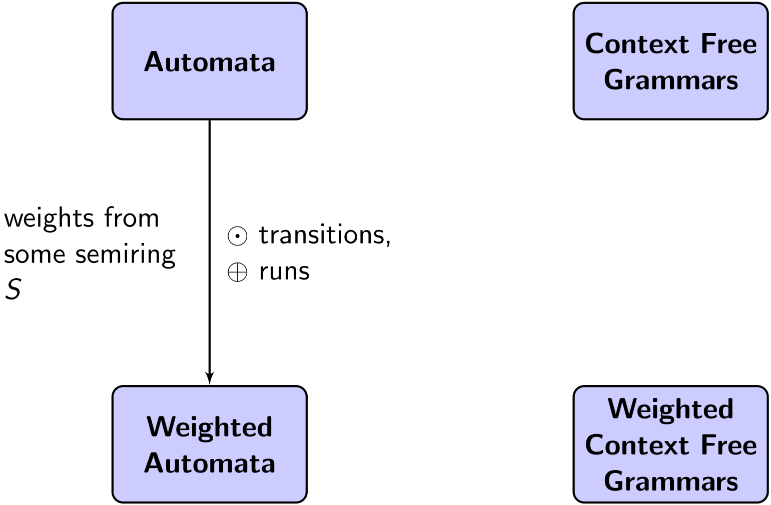
Automata

Weighted Automata

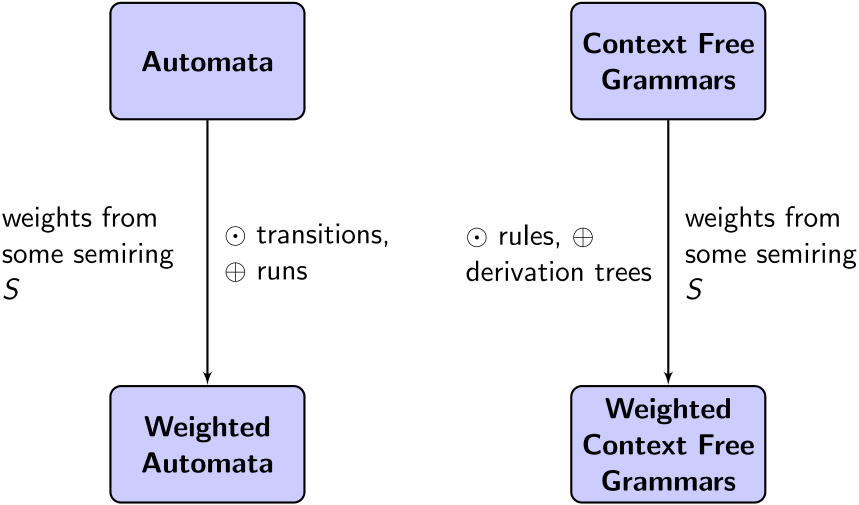
WCFG - nonlinear extension:



WCFG - nonlinear extension:



WCFG - nonlinear extension:



Automata

weights from some semiring S

$\odot$ transitions, $\oplus$ runs

Weighted Automata

Context Free Grammars

$\odot$ rules, $\oplus$ derivation trees

weights from some semiring S

Weighted Context Free Grammars

WCFG - nonlinear extension

Weighted Context Free Grammar:

WCFG - nonlinear extension

Weighted Context Free Grammar:

- Always in Greibach Normal Form
- left most derivation tree

WCFG - nonlinear extension

Weighted Context Free Grammar:

- $\boxed{\text{Always in Greibach Normal Form}}$
- $\boxed{\text{left most derivation tree}}$

$S \to aAB$ 1
$A \to b$ 1 $|bA$ 2
$B \to b$ 1 $|bB$ 3

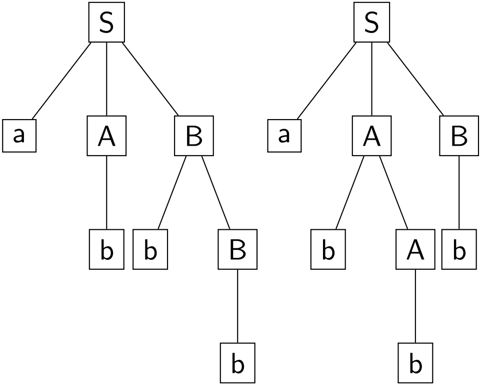
WCFG - nonlinear extension:

$S \rightarrow aAB$ 1

$A \rightarrow b$ 1 $|bA$ 2

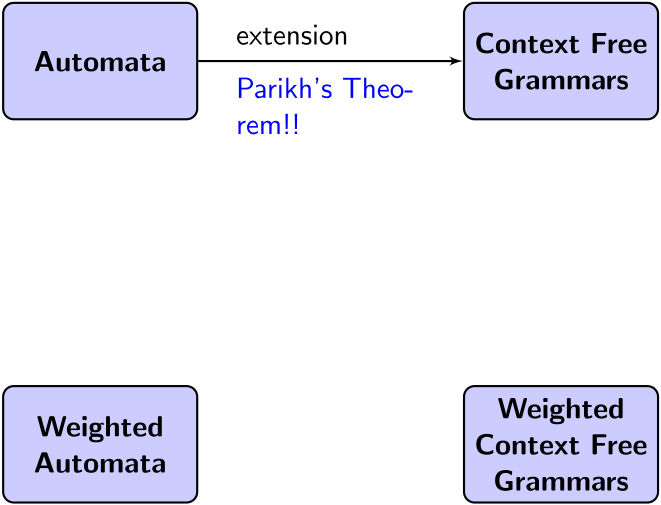
$B \rightarrow b$ 1 $|bB$ 3

Consider *abbb*:



$\boxed{\text{Output} = 1.1.3.1 + 1.2.1.1 = 5}$

WCFG - nonlinear extension:



Automata

extension

Parikh's Theorem!!

Context Free Grammars

Weighted Automata

Weighted Context Free Grammars

Parikh's Theorem:

Parikh Map : $Pk(w) \overset{\text{def}}{=} (\#a_w, \#b_w, \ldots)$

Parikh's Theorem

For every context-free grammar G, there is a regular language R such that $Pk(L(G)) = Pk(L(R))$.

Parikh's Theorem:

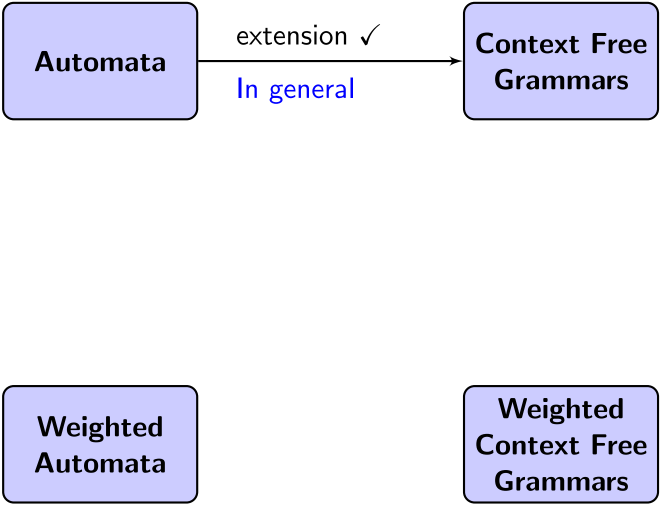
Parikh Map : $Pk(w) \overset{\text{def}}{=} (\#a_w, \#b_w, \ldots)$

Parikh's Theorem

For every context-free grammar G, there is a regular language R such that $Pk(L(G)) = Pk(L(R))$.

Corollary: For every context-free grammar G on unary alphabet, there is a regular language R such that $L(G) = L(R)$.

WCFG - nonlinear extension:



Automata

extension ✓

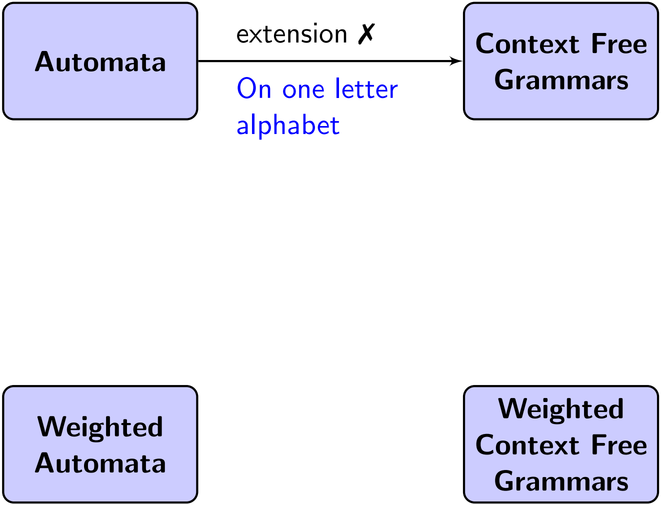
In general

Context Free Grammars

Weighted Automata

Weighted Context Free Grammars

WCFG - nonlinear extension:



Automata

extension ✗

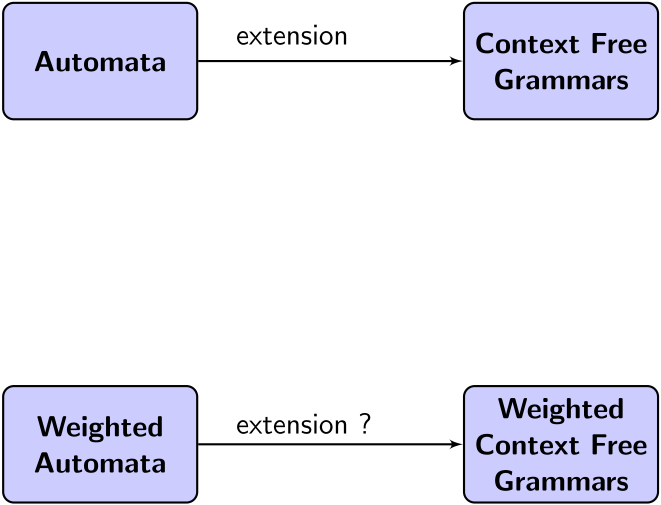
On one letter alphabet

Context Free Grammars

Weighted Automata

Weighted Context Free Grammars

WCFG - nonlinear extension:



Parikh's Theorem for WCFG :

Parikh image for a WCFG (G, W) **:**

$$Pk[\![G]\!]_W(u) \overset{\text{def}}{=} \bigoplus_{u' \in [u]_{Pk}} [\![G]\!]_W(u')$$

Parikh's Theorem for WCFG :

Parikh image for a WCFG (G, W) **:**

$$Pk[\![G]\!]_W(u) \overset{\text{def}}{=} \bigoplus_{u' \in [u]_{Pk}} [\![G]\!]_W(u')$$

Weighted Parikh's Theorem [BGV]

For every weighted context-free grammar (G,W) over an idempotent, commutative semiring, Parikh's theorem holds.

Parikh's Theorem for WCFG :

Parikh image for a WCFG (G, W) **:**

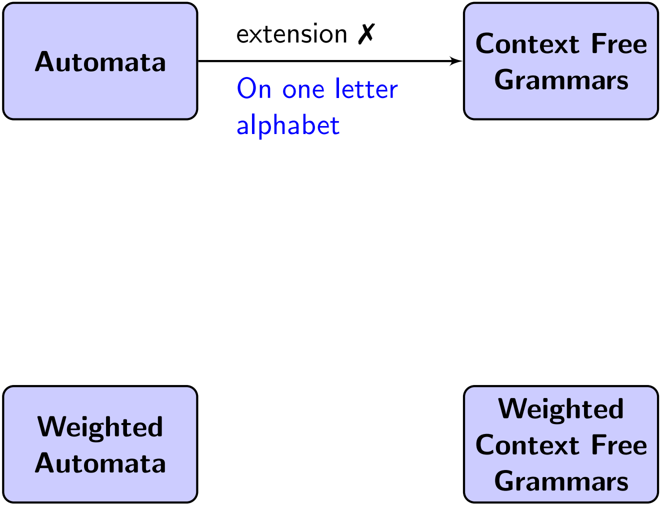
$$Pk[\![G]\!]_W(u) \overset{\text{def}}{=} \bigoplus_{u' \in [u]_{Pk}} [\![G]\!]_W(u')$$

Weighted Parikh's Theorem [BGV]

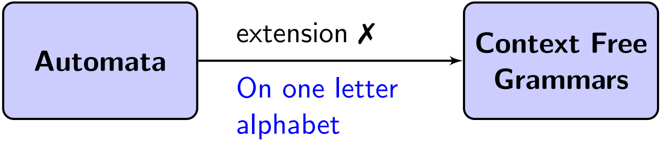
For every weighted context-free grammar (G,W) over an idempotent, commutative semiring, Parikh's theorem holds.

Note: Idempotent is really necessary!!

WCFG - nonlinear extension:



WCFG - nonlinear extension:

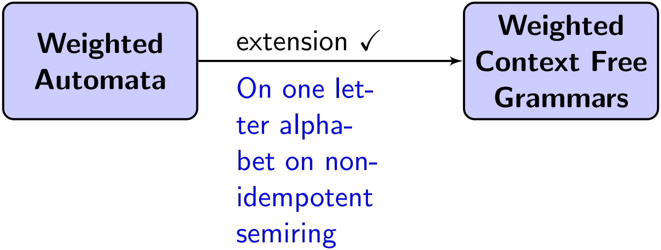


Automata

extension ✗

On one letter alphabet

Context Free Grammars



Weighted Automata

extension ✓

On one letter alphabet on non-idempotent semiring

Weighted Context Free Grammars

WCFG -nonlinear extension:

Consider the grammar:
$S \rightarrow aSS\ 1\ |a\ 1$

WCFG -nonlinear extension:

Consider the grammar:
$S \rightarrow aSS\ 1\ |a\ 1$

This grammar only produces words of the form a^{2k+1}.

WCFG -nonlinear extension:

Consider the grammar:
$S \rightarrow aSS\ 1\ |a\ 1$

This grammar only produces words of the form a^{2k+1}.

The number of derivation trees of the word a^{2k+1} is C_k, k-th Catalan Number.

WCFG -nonlinear extension:

Consider the grammar:
$S \rightarrow aSS\ 1\ |a\ 1$

This grammar only produces words of the form a^{2k+1}.

The number of derivation trees of the word a^{2k+1} is C_k, k-th Catalan Number.$\Rightarrow$ weight of $(a^{2k+1}) = C_k$.

WCFG -nonlinear extension:

Consider the grammar:
$S \rightarrow aSS\ 1\ |a\ 1$

This grammar only produces words of the form a^{2k+1}.

The number of derivation trees of the word a^{2k+1} is C_k, k-th Catalan Number.$\Rightarrow$ weight of $(a^{2k+1}) = C_k$.

It can be shown that generating function of Catalan number is not rational.

WCFG -nonlinear extension:

Consider the grammar:
$S \rightarrow aSS\ 1\ |a\ 1$

This grammar only produces words of the form a^{2k+1}.

The number of derivation trees of the word a^{2k+1} is C_k, k-th Catalan Number.$\Rightarrow$ weight of $(a^{2k+1}) = C_k$.

It can be shown that generating function of Catalan number is not rational.
$\Rightarrow$ There is no equivalent WFA accepting the same weighted-language.

WCFG -nonlinear extension:

Consider the grammar:
$S \rightarrow aSS\ 1\ |a\ 1$

This grammar only produces words of the form a^{2k+1}.

The number of derivation trees of the word a^{2k+1} is C_k, k-th Catalan Number.$\Rightarrow$ weight of $(a^{2k+1}) = C_k$.

It can be shown that generating function of Catalan number is not rational.
$\Rightarrow$ There is no equivalent WFA accepting the same weighted-language.

$\boxed{\text{Now, you can really believe, it is an extension!!}}$

Outline

1. Weighted Automata
2. Hankel Matrix
3. Ambiguity
4. Universality with Ambiguity
5. Introduction to Weighted Context-Free Grammar
6. Learning WCFG
7. Properties of WCFG

Learning WCFG:

We know that the Hankel matrix for functions $f : \Sigma^* \to \mathbb{R}$ can be used to characterise functions recognised by weighted automata.

Learning WCFG:

We know that the Hankel matrix for functions $f : \Sigma^* \rightarrow \mathbb{R}$ can be used to characterise functions recognised by weighted automata. *Can we do something similar for WCFG?*

Learning WCFG:

We know that the Hankel matrix for functions $f : \Sigma^* \to \mathbb{R}$ can be used to characterise functions recognised by weighted automata. *Can we do something similar for WCFG?*

The same kind of Hankel Matrix does not contain enough information, i.e. it can have infinite rank though recognised by a WCFG.

Learning WCFG:

We know that the Hankel matrix for functions $f : \Sigma^* \to \mathbb{R}$ can be used to characterise functions recognised by weighted automata. *Can we do something similar for WCFG?*

The same kind of Hankel Matrix does not contain enough information, i.e. it can have infinite rank though recognised by a WCFG.

Bailly, Carreras, Luque, and Quattoni presented a similar Hankel-like theorem in their paper.

Learning WCFG:

We know that the Hankel matrix for functions $f : \Sigma^* \to \mathbb{R}$ can be used to characterise functions recognised by weighted automata. *Can we do something similar for WCFG?*

The same kind of Hankel Matrix does not contain enough information, i.e. it can have infinite rank though recognised by a WCFG.

Bailly, Carreras, Luque, and Quattoni presented a similar Hankel-like theorem in their paper.

But, the idea is WRONG!!

Learning WCFG:

The idea was following:

Learning WCFG:

The idea was following:
We consider functions $f : O \times I \to \mathbb{R}$, where $O \in \Sigma^* \times \Sigma^*$ and $I \in \Sigma^+$.
$f(\langle x; z \rangle, y) = [\![G]\!]_W(xyz)$.

Learning WCFG:

The idea was following:
We consider functions $f : O \times I \to \mathbb{R}$, where $O \in \Sigma^* \times \Sigma^*$ and $I \in \Sigma^+$.
$f(\langle x; z\rangle, y) = [\![G]\!]_W(xyz)$.

To compute this function, they defined two functions:
Inside Function: $\overline{\beta}_G(i \Rightarrow^* y)$[Intuitively, denotes the weight of deriving y from a non-terminal i]
Outside Function: $\overline{\alpha}_G(x; i; z)$[Intuitively denotes the weight of derivation of the context$\langle x; z\rangle$]

Learning WCFG:

Hence, $[\![G]\!]_W(xyz) = \sum_{i \in V} \overline{\alpha}_G(x; i; z)\overline{\beta}_G(i \Rightarrow^* y)$.

Learning WCFG:

Hence, $[\![G]\!]_W(xyz) = \sum_{i \in V} \overline{\alpha}_G(x; i; z)\overline{\beta}_G(i \Rightarrow^* y)$.

Using, this idea they constructed the hankel matrix for WCFG like the following:

Learning WCFG:

Hence, $[\![G]\!]_W(xyz) = \sum_{i\in V} \overline{\alpha}_G(x;i;z)\overline{\beta}_G(i \Rightarrow^* y)$.

Using, this idea they constructed the hankel matrix for WCFG like the following:

$$H_{O\times I} = \begin{array}{cc} & \begin{array}{ccc} & & y \end{array} \\ \langle x;z\rangle & \left(\begin{array}{ccc} & & . \\ & & . \\ . & . & [\![G]\!]_W(xyz) \\ & & \end{array}\right) \end{array}$$

Learning WCFG:

Hence, $[\![G]\!]_W(xyz) = \sum_{i \in V} \overline{\alpha}_G(x;i;z)\overline{\beta}_G(i \Rightarrow^* y)$.

Using, this idea they constructed the hankel matrix for WCFG like the following:

$$H_{O\times I} = \begin{array}{cc} & y \\ \langle x;z\rangle & \begin{pmatrix} & & . & \\ & & . & \\ . & . & [\![G]\!]_W(xyz) & \\ & & & \end{pmatrix} \end{array}$$

This matrix has finite rank. Surprisingly, their following theorem says, this is enough information to learn the WCFG.

Learning WCFG

Theorem [BCLQ]

Given a complete basis for the Hankel Matrix defined above for a function f recognized by a WCFG, we can effectively construct the WCFG from that basis.

Learning WCFG

Theorem [BCLQ]

Given a complete basis for the Hankel Matrix defined above for a function f recognized by a WCFG, we can effectively construct the WCFG from that basis.

But this idea is WRONG!!!

Learning WCFG

Theorem [BCLQ]

Given a complete basis for the Hankel Matrix defined above for a function f recognized by a WCFG, we can effectively construct the WCFG from that basis.

But this idea is WRONG!!!

We now give a counter-example.

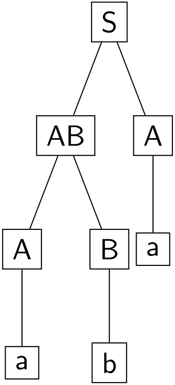
Learning WCFG:

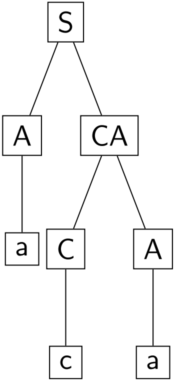
Precisely, the wrong claim is that for any $f : (\Sigma^* \times \Sigma^*) \times \Sigma^+ \rightarrow \mathbb{R}$, one can construct a weighted context-free grammar computing f with the number of non-terminals being the rank of H_f.

Learning WCFG:

Precisely, the wrong claim is that for any $f : (\Sigma^* \times \Sigma^*) \times \Sigma^+ \to \mathbb{R}$, one can construct a weighted context-free grammar computing f with the number of non-terminals being the rank of H_f.

We start from the function $f : \mathit{Tree}(\Sigma) \to \mathbb{R}$ assigning 1 to the following two trees, and 0 to any other tree.





Learning WCFG

Clearly, there exists a WCFG with 6 non-terminals defining the function f.

Learning WCFG

Clearly, there exists a WCFG with 6 non-terminals defining the function f.

But if we construct the Hankel Matrix for the function $f : O \times I \to \mathbb{R}$ for this, we will see it has rank 5.

Learning WCFG

Clearly, there exists a WCFG with 6 non-terminals defining the function f.

But if we construct the Hankel Matrix for the function $f : O \times I \to \mathbb{R}$ for this, we will see it has rank 5.

But, no WCFG with 5 non-terminals accept this language.

Learning WCFG:

What went wrong??

Learning WCFG:

What went wrong??

There exists a natural extension for Fliess' theorem for Weighted tree automata by Bozapalidis and Louscou-Bozapalidou([BL83]).

Learning WCFG:

What went wrong??

There exists a natural extension for Fliess' theorem for Weighted tree automata by Bozapalidis and Louscou-Bozapalidou([BL83]). Consider a function $f : \mathit{Tree}(\Sigma) \rightarrow \mathbb{R}$. A context is a tree over the signature $\Sigma \cup \square(0)$ with the restriction that $\square$ occurs only once.

Learning WCFG:

What went wrong??

There exists a natural extension for Fliess' theorem for Weighted tree automata by Bozapalidis and Louscou-Bozapalidou([BL83]).
Consider a function $f : \mathit{Tree}(\Sigma) \rightarrow \mathbb{R}$. A context is a tree over the signature $\Sigma \cup \square(0)$ with the restriction that $\square$ occurs only once.

A context c and a tree t, yield a tree $c[t]$, where we substitute the leaf $\square$ in c by t.

Naturally the Hankel Matrix $H_f \in \mathbb{R}^{\mathit{Context}(\Sigma) \times \mathit{Tree}(\Sigma)}$ such that $H_f(c, t) = f(c[t])$ can be defined and the Fliess' theorem can be extended over this.

Learning WCFG

Now, consider the previous language. The tree hankel matrix will correctly have rank 6 for the function f, but the WCFG hankel matrix will have rank 5.

Learning WCFG

Now, consider the previous language. The tree hankel matrix will correctly have rank 6 for the function f, but the WCFG hankel matrix will have rank 5.Why?

Learning WCFG

Now, consider the previous language. The tree hankel matrix will correctly have rank 6 for the function f, but the WCFG hankel matrix will have rank 5.Why?

This is beacuse the row for the context $\langle a; a \rangle$ has value 1 for b and c according to Baily et al's Hankel Matrix.

Learning WCFG

Now, consider the previous language. The tree hankel matrix will correctly have rank 6 for the function f, but the WCFG hankel matrix will have rank 5.Why?

This is beacuse the row for the context $\langle a; a\rangle$ has value 1 for b and c according to Baily et al's Hankel Matrix.

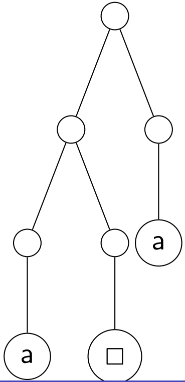
But, for the tree hankel matrix it will have two seperate contexts:

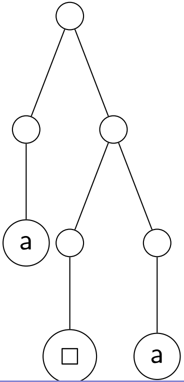
Learning WCFG

Now, consider the previous language. The tree hankel matrix will correctly have rank 6 for the function f, but the WCFG hankel matrix will have rank 5.Why?

This is beacuse the row for the context $\langle a; a \rangle$ has value 1 for b and c according to Baily et al's Hankel Matrix.

But, for the tree hankel matrix it will have two seperate contexts:





Outline

1. Weighted Automata
2. Hankel Matrix
3. Ambiguity
4. Universality with Ambiguity
5. Introduction to Weighted Context-Free Grammar
6. Learning WCFG
7. Properties of WCFG

WA & LRS:

Let's come back to LRS again:

WA & LRS:

Let's come back to LRS again:

Linear Recurrence System: Each term of a sequence is a linear function of earlier terms in the sequence.

$$\underbrace{\left\{\begin{array}{l} f(n) = f(n-1) + g(n-1) \\ g(n) = f(n-1) \\ f(0) = 0 \\ g(0) = 1 \end{array}\right\}}_{\boxed{\text{Fibonacci}}} \Leftrightarrow \left\{\begin{array}{l} f(n) = f(n-1) + f(n-2) \\ f(0) = 0 \\ f(1) = 1 \end{array}\right\}$$

WA & LRS:

Consider $\Sigma = \{a\}$

WA & LRS:

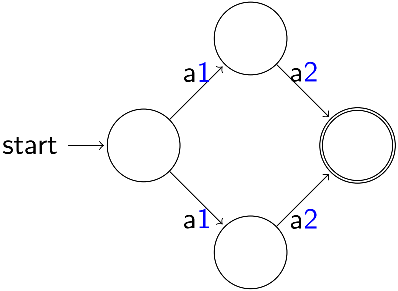
Consider $\Sigma = \{a\}$

$f : \Sigma^* \to \mathbb{R} \Rightarrow f' : \mathbb{N} \to \mathbb{R}$ $\boxed{f'(n) = f(a^n)}$

WA & LRS:

Consider $\Sigma = \{a\}$

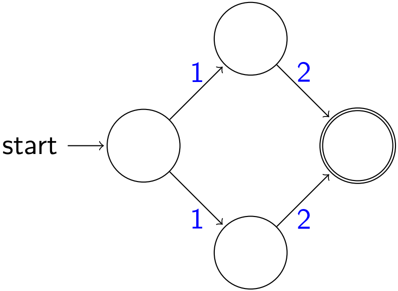
$f : \Sigma^* \rightarrow \mathbb{R} \Rightarrow f' : \mathbb{N} \rightarrow \mathbb{R}$ $\boxed{f'(n) = f(a^n)}$



WA & LRS:

Consider $\Sigma = \{a\}$

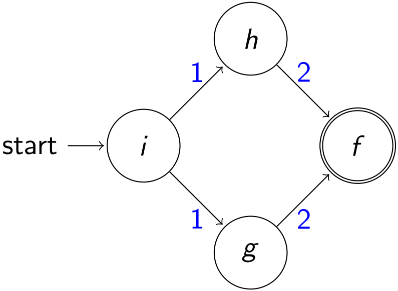
$f : \Sigma^* \to \mathbb{R} \Rightarrow f' : \mathbb{N} \to \mathbb{R}$ $\boxed{f'(n) = f(a^n)}$



WA & LRS:

Consider $\Sigma = \{a\}$

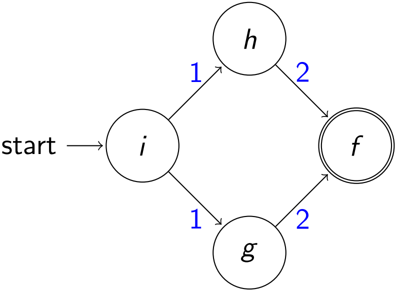
$f : \Sigma^* \to \mathbb{R} \Rightarrow f' : \mathbb{N} \to \mathbb{R}$ $\boxed{f'(n) = f(a^n)}$



WA & LRS:

Consider $\Sigma = \{a\}$

$f : \Sigma^* \to \mathbb{R} \Rightarrow f' : \mathbb{N} \to \mathbb{R}$ $\boxed{f'(n) = f(a^n)}$

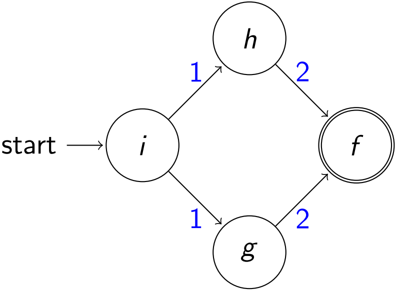


$$\left\{ \begin{array}{l} f(n) = 2.g(n-1) + 2.h(n-1) \\ g(n) = i(n-1) \\ h(n) = i(n-1) \end{array} \right\}$$

WA & LRS:

Consider $\Sigma = \{a\}$

$f : \Sigma^* \to \mathbb{R} \Rightarrow f' : \mathbb{N} \to \mathbb{R}$ $\boxed{f'(n) = f(a^n)}$



$$\left\{ \begin{array}{l} f(n) = 2.g(n-1) + 2.h(n-1) \\ g(n) = i(n-1) \\ h(n) = i(n-1) \end{array} \right\}$$

Intuitively, counting the number of paths!!

WCFG & Recursions:

WFA on one letter alphabet $\Rightarrow$ Linear Recurrence System

WCFG & Recursions:

WFA on one letter alphabet $\Rightarrow$ Linear Recurrence System

Intuition was to count the number of accepting runs!!

WCFG & Recursions:

WFA on one letter alphabet ⇒ Linear Recurrence System

Intuition was to count the number of accepting runs!!

WCFG ⇒ ??

WCFG & Recursions:

WFA on one letter alphabet $\Rightarrow$ Linear Recurrence System

Intuition was to count the number of accepting runs!!

WCFG $\Rightarrow$??

Now, Intuition is to count the number of derivation trees!!

WCFG & Recursions:

WFA on one letter alphabet ⇒ Linear Recurrence System

Intuition was to count the number of accepting runs!!

WCFG ⇒ ??

Now, Intuition is to count the number of derivation trees!!

$S \rightarrow aA_1\ 3\ |aA_3A_4\ 2$

WCFG & Recursions:

WFA on one letter alphabet $\Rightarrow$ Linear Recurrence System

Intuition was to count the number of accepting runs!!

WCFG $\Rightarrow$??

Now, Intuition is to count the number of derivation trees!!

$S \rightarrow aA_1\ 3\ |aA_3A_4\ 2$

$S(n) = 3.A_1(n-1) + 2.A_3 * A_4(n-1)$, where
$f * g(k) = \sum_{i=0}^{k} f(i).g(k-i)$

Same idea as in Catalan number!!

WCFG & Recursions:

WFA on one letter alphabet ⇒ Linear Recurrence System

Intuition was to count the number of accepting runs!!

WCFG ⇒ ??

Now, Intuition is to count the number of derivation trees!!

$S \to aA_1\ 3\ |aA_3A_4\ 2$

$S(n) = 3.A_1(n-1) + 2.A_3 * A_4(n-1)$, where
$f * g(k) = \sum_{i=0}^{k} f(i).g(k-i)$

Same idea as in Catalan number!!

WCFG ⇒ Linear Recurrence System with finitely many Cauchy product.

WCFG & mathematical characterization:

Function recognized by WFA on one letter alphabet⇒ Rational function

WCFG & mathematical characterization:

Function recognized by WFA on one letter alphabet$\Rightarrow$ Rational function

Function recognized by WCFG on one letter alphabet $\Rightarrow$ Something special??

WCFG & mathematical characterization:

Function recognized by WFA on one letter alphabet⇒ Rational function

Function recognized by WCFG on one letter alphabet ⇒ Something special??

Formally, let p_k = weight of the word a^k in G. Can we characterize the power series $P(x) = \sum_{k=0}^{\infty} p_k x^k$?

WCFG & mathematical characterization:

Function recognized by WFA on one letter alphabet⇒ Rational function

Function recognized by WCFG on one letter alphabet ⇒ Something special??

Formally, let $p_k =$ weight of the word a^k in G. Can we characterize the power series $P(x) = \sum_{k=0}^{\infty} p_k x^k$?

Notice that this is the generating function of the given weighted grammar.

WCFG & mathematical characterization:

Chomsky–Schützenberger Enumeration Theorem

If L is a context-free language admitting an unambiguous context-free grammar, and $a_k := |L \cap \Sigma^k|$ is the number of words of length k in L, then $G(x) = \sum_{k=0}^{\infty} a_k x^k$ is a power series over $\mathbb{N}$ that is algebraic over $\mathbb{Q}(x)$.

WCFG & mathematical characterization:

Chomsky–Schützenberger Enumeration Theorem

If L is a context-free language admitting an unambiguous context-free grammar, and $a_k := |L \cap \Sigma^k|$ is the number of words of length k in L, then $G(x) = \sum_{k=0}^{\infty} a_k x^k$ is a power series over $\mathbb{N}$ that is algebraic over $\mathbb{Q}(x)$.

Consider any unary WCFG G on semiring $\mathbb{N}$ with all weights 1. For every rule, we replace the terminal a with a new terminal and produce a different grammar G' on a large alphabet.

WCFG & mathematical characterization:

Chomsky–Schützenberger Enumeration Theorem

If L is a context-free language admitting an unambiguous context-free grammar, and $a_k := |L \cap \Sigma^k|$ is the number of words of length k in L, then $G(x) = \sum_{k=0}^{\infty} a_k x^k$ is a power series over $\mathbb{N}$ that is algebraic over $\mathbb{Q}(x)$.

Consider any unary WCFG G on semiring $\mathbb{N}$ with all weights 1. For every rule, we replace the terminal a with a new terminal and produce a different grammar G' on a large alphabet.

It can be shown that, G' is unambiguous and ambiguity of a^k in $G=$ number of k-length words in G'.

WCFG & mathematical characterization:

Chomsky–Schützenberger Enumeration Theorem

If L is a context-free language admitting an unambiguous context-free grammar, and $a_k := |L \cap \Sigma^k|$ is the number of words of length k in L, then $G(x) = \sum_{k=0}^{\infty} a_k x^k$ is a power series over $\mathbb{N}$ that is algebraic over $\mathbb{Q}(x)$.

Consider any unary WCFG G on semiring $\mathbb{N}$ with all weights 1. For every rule, we replace the terminal a with a new terminal and produce a different grammar G' on a large alphabet.

It can be shown that, G' is unambiguous and ambiguity of a^k in $G =$ number of k-length words in G'.

What happens if all the weights are not 1?

WCFG & mathematical characterization:

Suppose some rule has weight $k \in \mathbb{N}$.

WCFG & mathematical characterization:

Suppose some rule has weight $k \in \mathbb{N}$. We will simply produce k- copies of the same rule with k- new terminals.

WCFG & mathematical characterization:

Suppose some rule has weight $k \in \mathbb{N}$. We will simply produce k- copies of the same rule with k- new terminals. $\boxed{\text{a huge alphabet!!}}$

WCFG & mathematical characterization:

Suppose some rule has weight $k \in \mathbb{N}$. We will simply produce k- copies of the same rule with k- new terminals. $\boxed{\text{a huge alphabet!!}}$

Now, weight of a^k in $G=$ number of k-length words in G'.

WCFG & mathematical characterization:

Suppose some rule has weight $k \in \mathbb{N}$. We will simply produce k- copies of the same rule with k- new terminals. $\boxed{\text{a huge alphabet!!}}$

Now, weight of a^k in $G=$ number of k-length words in G'.
$\boxed{\text{Apply Enumeration Theorem!!}}$

WCFG & mathematical characterization:

Suppose some rule has weight $k \in \mathbb{N}$. We will simply produce k- copies of the same rule with k- new terminals. $\boxed{\text{a huge alphabet!!}}$

Now, weight of a^k in $G=$ number of k-length words in G'.
$\boxed{\text{Apply Enumeration Theorem!!}}$

Corollary

Given a WCFG on $\mathbb{N}$ on a unary alphabet, the generating function $P(x) = \sum_{k=0}^{\infty} p_k x^k$ is algebraic over $\mathbb{Q}(x)$.

Conclusion

Further Questions:

- How to effectively learn a Weighted Context-Free Grammar?
- Better mathematical characterizations for functions realized by WCFG?

References

- Raphaël Bailly, Xavier Carreras, Franco M. Luque, and Ariadna Quattoni.
  Unsupervised spectral learning of WCFG as low-rank matrix completion.

- Vijay Bhattiprolu, Spencer Gordon, and Mahesh Viswanathan.
  Extending parikh's theorem to weighted and probabilistic context-free grammars.
  Lecture Notes in Computer Science.

- Symeon Bozapalidis and Olympia Louscou-Bozapalidou.
  The rank of a formal tree power series.
  Theor. Comput. Sci., 27:211–215, 1983.

- Nathanael Fijalkow.
  Blog-post on angluin's style learning for weighted automata.

- Nathanael Fijalkow.
  Blog-post on fliess' theorem for minimising weighted automata.

Thank you!!

AND THAT'S MY LAST SLIDE. ANY COMMENTS?
YOU STOLE AN HOUR OF MY LIFE. SOMETHING INSIDE ME DIED. I WILL NEVER HAVE ANOTHER GOOD DAY.
ZZZZ
I WENT IN WITH LOW EX-PECTA-TIONS.
THEY CAN'T HURT YOU IF YOU'RE ALREADY DEAD.
Dilbert.com DilbertCartoonist@gmail.com
12-7-11 ©2011 Scott Adams, Inc./Dist. by Universal Uclick