Weighted automata with pebbles and weighted FO logic with transitive closures

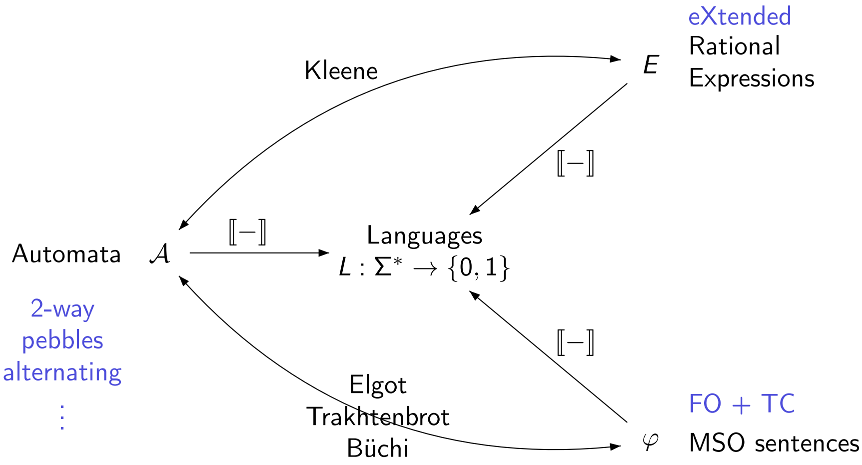
Paul Gastin

Benedikt Bollig, Benjamin Monmege, Marc Zeitoun
LSV, ENS Cachan, CNRS, INRIA.

Dagstuhl
Dec. 13-17, 2010

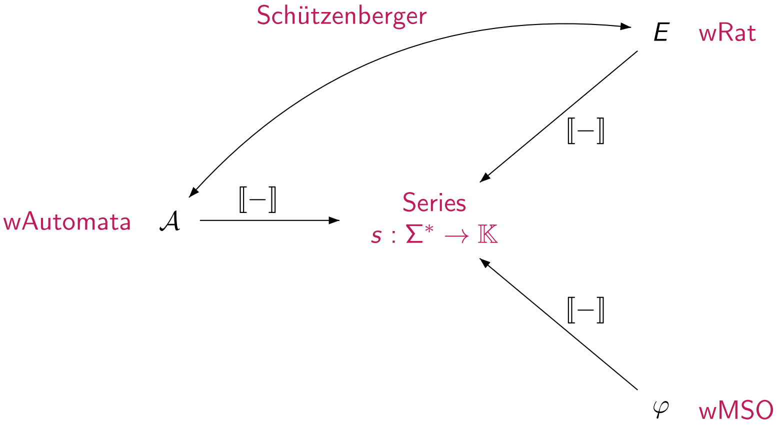
Preliminary version at ICALP'10

Motivation: The Paradise for weights



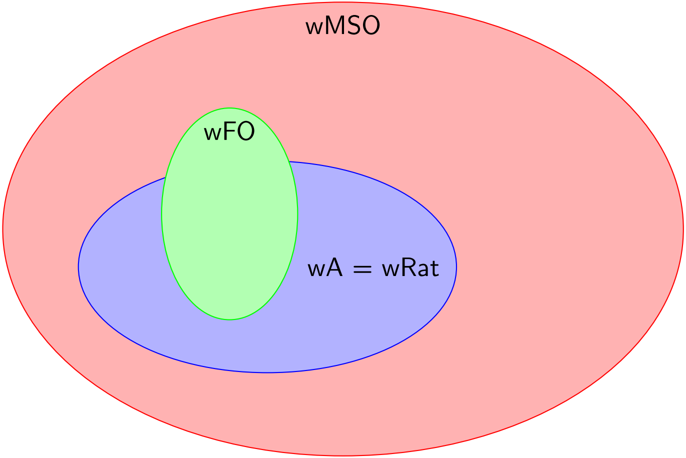
Boolean: $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$

Motivation: The Paradise for weights

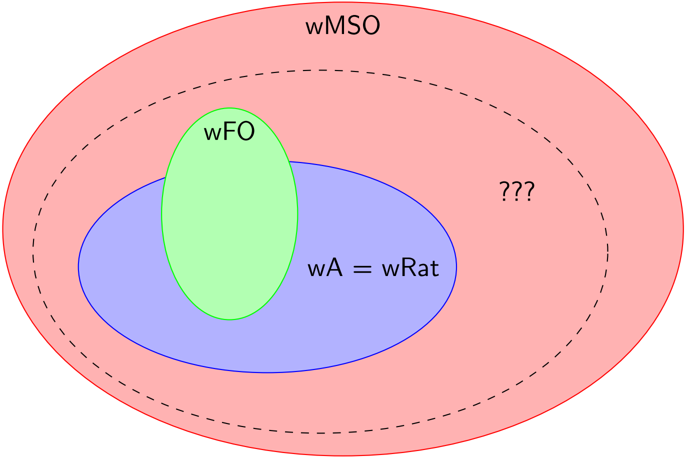


Quantitative: $\mathbb{K} = (K, +, \times, 0, 1)$

Expressivity in weighted setting



Expressivity in weighted setting



Find a robust class containing both wFO and wAutomata.

Weighted automata

- Transitions carry weights from a semiring $\mathbb{K}$: $\mu : \Sigma \to K^{Q\times Q}$.

- Weight of a run on $w = a_1 a_2 \cdots a_n$: product in the semiring.

$$\text{weight}(p_0 \xrightarrow{k_1 a_1} p_1 \xrightarrow{k_2 a_2} \cdots \xrightarrow{k_n a_n} p_n) = k_1 k_2 \cdots k_n$$

- Value of a word: sum of all weights of runs on this word.

$$[\![\mathcal{A}]\!](w) = \sum_{\rho \text{ run of } \mathcal{A} \text{ on } w} \text{weight}(\rho) = \lambda \cdot \mu(w) \cdot \gamma$$

Weighted automata

- Transitions carry weights from a semiring $\mathbb{K}$: $\mu : \Sigma \to K^{Q \times Q}$.

- Weight of a run on $w = a_1 a_2 \cdots a_n$: product in the semiring.

$$\text{weight}(p_0 \xrightarrow{k_1 a_1} p_1 \xrightarrow{k_2 a_2} \cdots \xrightarrow{k_n a_n} p_n) = k_1 k_2 \cdots k_n$$

- Value of a word: sum of all weights of runs on this word.

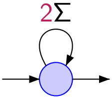
$$[\![\mathcal{A}]\!](w) = \sum_{\rho \text{ run of } \mathcal{A} \text{ on } w} \text{weight}(\rho) = \lambda \cdot \mu(w) \cdot \gamma$$

Example: Semirings: $\mathbb{K} = (K, +, \times, 0, 1)$

- $\mathbb{B} = (\{0,1\}, \vee, \wedge, 0, 1)$ — Boolean
- $\mathbb{P} = (\mathbb{R}^+, +, \times, 0, 1)$ — Probabilistic
- $\mathbb{N} = (\mathbb{N}, +, \times, 0, 1)$ — Natural
- $\mathbb{T} = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$ — Tropical

Examples of weighted automata

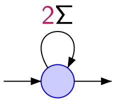
- Alphabet Σ, on $(\mathbb{N}, +, \times, 0, 1)$



$[\![\mathcal{A}]\!](u) = 2^{|u|}$ (deterministic)

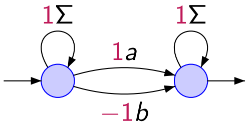
Examples of weighted automata

- Alphabet Σ, on $(\mathbb{N}, +, \times, 0, 1)$



$[\![\mathcal{A}]\!](u) = 2^{|u|}$ (deterministic)

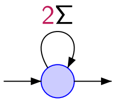
- Alphabet $\Sigma = \{a, b\}$, on $(\mathbb{Z}, +, \times, 0, 1)$



$[\![\mathcal{A}]\!](u) = |u|_a - |u|_b$

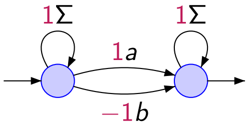
Examples of weighted automata

- Alphabet Σ, on $(\mathbb{N}, +, \times, 0, 1)$



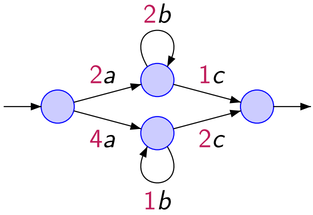
$[\![\mathcal{A}]\!](u) = 2^{|u|}$ (deterministic)

- Alphabet $\Sigma = \{a, b\}$, on $(\mathbb{Z}, +, \times, 0, 1)$



$[\![\mathcal{A}]\!](u) = |u|_a - |u|_b$

- Alphabet $\{a, b, c\}$, on $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$



$[\![\mathcal{A}]\!](ab^n c) = \min(3 + 2n, 6 + n)$

Weighted automata cannot compute large weights

Remark

$\mathcal{A} = (Q, \mu)$ weighted automaton on $\mathbb{N}$. There exists M such that

$$[\![\mathcal{A}]\!](u) = O(M^{|u|}).$$

- There are $|Q|^{|u|+1}$ runs on $u = a_1 a_2 \cdots a_n$,

$$\rho = p_0 \xrightarrow{k_1 a_1} p_1 \xrightarrow{k_2 a_2} \cdots \xrightarrow{k_n a_n} p_n$$

- The weight of a run is exponential in $|u|$:

$$\mathrm{weight}(\rho) = k_1 k_2 \cdots k_n \leq (\max\{\mu(a)_{p,q} \mid a \in \Sigma \text{ and } p, q \in Q\})^{|u|}.$$

Weighted MSO

Definition: Syntax of wMSO

$$\varphi ::= k \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x\, \varphi \mid \forall x\, \varphi \mid \exists X\, \varphi \mid \forall X\, \varphi$$

where $k \in K$, $a \in \Sigma$, x, y are first-order variables, X is a set variable.

Definition: Semantics

- A formula φ without free variables defines a mapping $[\![\varphi]\!] : \Sigma^+ \to K$.
- First order variables are interpreted as positions in the word.
- $P_a(x)$ means "position x carries an a".
- $x \leq y$ means "position x is before position y".

Weighted MSO

Definition: Syntax of wMSO

$$\varphi ::= k \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x\, \varphi \mid \forall x\, \varphi \mid \exists X\, \varphi \mid \forall X\, \varphi$$

where $k \in K$, $a \in \Sigma$, x, y are first-order variables, X is a set variable.

Definition: Semantics

- A formula φ without free variables defines a mapping $[\![\varphi]\!] : \Sigma^+ \to K$.
- First order variables are interpreted as positions in the word.
- $P_a(x)$ means "position x carries an a".
- $x \leq y$ means "position x is before position y".
- $[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] + [\![\varphi_2]\!]$ and $[\![\varphi_1 \wedge \varphi_2]\!] = [\![\varphi_1]\!] \times [\![\varphi_2]\!]$.

  Remember: $\mathbb{B} = (\{0,1\}, \vee, \wedge, 0, 1)$ and $\mathbb{K} = (K, +, \times, 0, 1)$.

Weighted MSO

Definition: Syntax of wMSO

$$\varphi ::= k \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x\, \varphi \mid \forall x\, \varphi \mid \exists X\, \varphi \mid \forall X\, \varphi$$

where $k \in K$, $a \in \Sigma$, x, y are first-order variables, X is a set variable.

Definition: Semantics

- A formula φ without free variables defines a mapping $[\![\varphi]\!] : \Sigma^+ \to K$.
- First order variables are interpreted as positions in the word.
- $P_a(x)$ means "position x carries an a".
- $x \leq y$ means "position x is before position y".
- $[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] + [\![\varphi_2]\!]$ and $[\![\varphi_1 \wedge \varphi_2]\!] = [\![\varphi_1]\!] \times [\![\varphi_2]\!]$.

  Remember: $\mathbb{B} = (\{0,1\}, \vee, \wedge, 0, 1)$ and $\mathbb{K} = (K, +, \times, 0, 1)$.
- $\exists x\, \varphi$ interpreted as a sum over all positions.
- $\forall x\, \varphi$ interpreted as a product over all positions.

wMSO: examples

- $[\![\exists x\ P_a(x)]\!](u) = \sum_{i \in \mathrm{pos}(u)} [\![P_a(x)]\!](u,i) = |u|_a$ recognizable

wMSO: examples

- $[\![\exists x\ P_a(x)]\!](u) = \sum_{i \in \mathrm{pos}(u)} [\![P_a(x)]\!](u,i) = |u|_a$ recognizable
- $[\![\forall y\ 2]\!](u) = \prod_{i \in \mathrm{pos}(u)} [\![2]\!](u,i) = 2^{|u|}$ recognizable

wMSO: examples

- $[\![\exists x\, P_a(x)]\!](u) = \sum_{i \in \mathrm{pos}(u)} [\![P_a(x)]\!](u,i) = |u|_a$ recognizable
- $[\![\forall y\, 2]\!](u) = \prod_{i \in \mathrm{pos}(u)} [\![2]\!](u,i) = 2^{|u|}$ recognizable
- $[\![\forall x\, \forall y\, 2]\!](u) = \prod_{i \in \mathrm{pos}(u)} [\![\forall y\, 2]\!](u,i) = (2^{|u|})^{|u|} = 2^{|u|^2}$. not recognizable

  w-Automata are not closed under universal quantification.

wMSO: examples

- $[\![\exists x\, P_a(x)]\!](u) = \sum_{i \in \mathrm{pos}(u)} [\![P_a(x)]\!](u,i) = |u|_a$ recognizable
- $[\![\forall y\, 2]\!](u) = \prod_{i \in \mathrm{pos}(u)} [\![2]\!](u,i) = 2^{|u|}$ recognizable
- $[\![\forall x \forall y\, 2]\!](u) = \prod_{i \in \mathrm{pos}(u)} [\![\forall y\, 2]\!](u,i) = (2^{|u|})^{|u|} = 2^{|u|^2}$. not recognizable

  w-Automata are not closed under universal quantification.

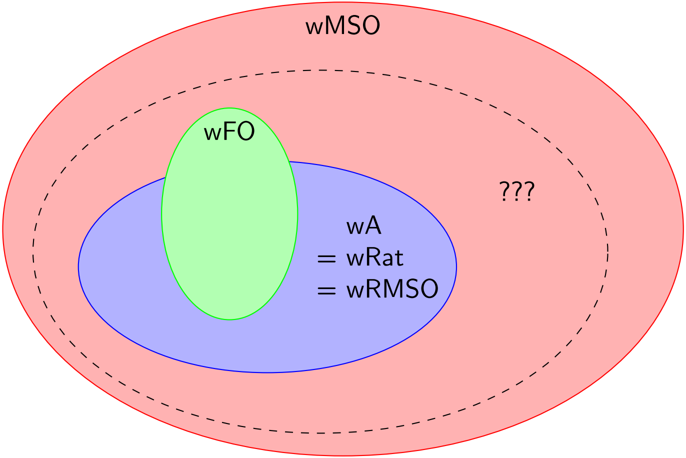
Theorem (Droste & Gastin'05)

$$\text{wAutomata} = \text{wRMSO}$$

wRMSO is a fragment of wMSO with

- $\forall X$ restricted to boolean formulae
- $\forall x$ restricted to $\bigvee \bigwedge$ of constants and boolean formulae

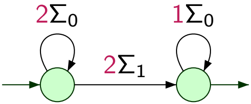
Extending instead of Restricting ?



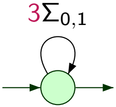
We aim at a robust class extending both wFO and wAutomata.

Nested automata (= 1-way pebble automata)

A 0-nested wA is a classical weighted automaton.



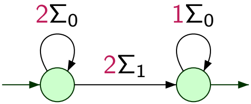
$[\![\mathcal{A}_1]\!](u) = 2^{i+1}$ if $u \in \Sigma_0^i \Sigma_1 \Sigma_0^*$

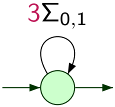


$[\![\mathcal{A}_2]\!](u) = 3^{|u|}$

Nested automata (= 1-way pebble automata)

A 0-nested wA is a classical weighted automaton.



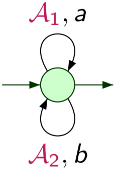


$[\![\mathcal{A}_1]\!](u) = 2^{i+1}$ if $u \in \Sigma_0^i \Sigma_1 \Sigma_0^*$

$[\![\mathcal{A}_2]\!](u) = 3^{|u|}$

Each transition $p \xrightarrow{a} q$ of an r-nested wA $\mathcal{A}$ calls an $(r-1)$-nested wA $\mathcal{A}_{p,a,q}$ with the current position i marked.

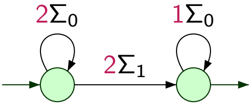
$\mathcal{A}_{p,a,q}$ restarts on (u, i) and computes the weight $p \xrightarrow{[\![\mathcal{A}_{p,a,q}]\!](u,i)a} q$.



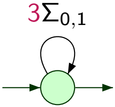
$[\![\mathcal{A}]\!](u) = 3^{|u||u|_b} \cdot 2^{\sum \mathrm{pos}(a,u)}$

Nested automata (= 1-way pebble automata)

A 0-nested wA is a classical weighted automaton.



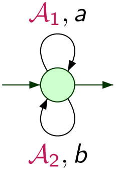
$[\![\mathcal{A}_1]\!](u) = 2^{i+1}$ if $u \in \Sigma_0^i \Sigma_1 \Sigma_0^*$



$[\![\mathcal{A}_2]\!](u) = 3^{|u|}$

Each transition $p \xrightarrow{a} q$ of an r-nested wA $\mathcal{A}$ calls an $(r-1)$-nested wA $\mathcal{A}_{p,a,q}$ with the current position i marked.

$\mathcal{A}_{p,a,q}$ restarts on (u, i) and computes the weight $p \xrightarrow{[\![\mathcal{A}_{p,a,q}]\!](u,i)a} q$.

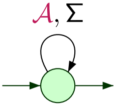


$[\![\mathcal{A}]\!](u) = 3^{|u||u|_b} \cdot 2^{\sum \mathrm{pos}(a,u)}$

An r-nested automaton does $1 + |u| + |u|^2 + \cdots + |u|^r$ 1-way runs on a word u.

Nested automata are closed under ∃ ∀

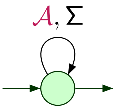
Proof: $\forall x\ \mathcal{A}(x)$



$$[\![\mathcal{B}]\!](u) = \prod_{i=1}^{|u|} [\![\mathcal{A}]\!](u, i)$$

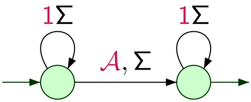
Nested automata are closed under ∃ ∀

Proof: $\forall x\ \mathcal{A}(x)$



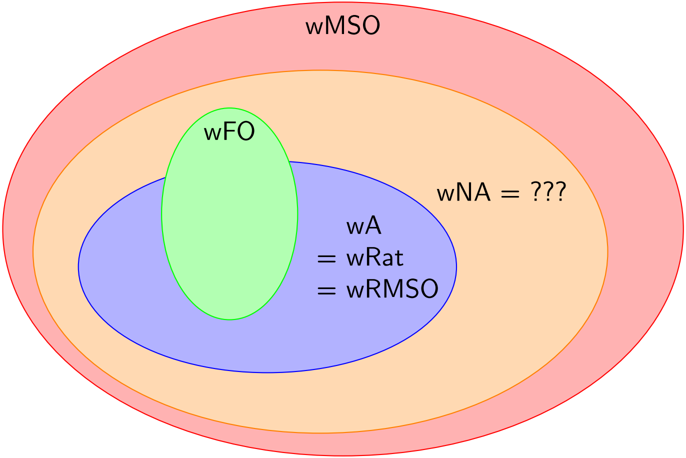
$$[\![\mathcal{B}]\!](u) = \prod_{i=1}^{|u|} [\![\mathcal{A}]\!](u, i)$$

Proof: $\exists x\ \mathcal{A}(x)$



$$[\![\mathcal{B}]\!](u) = \sum_{i=1}^{|u|} [\![\mathcal{A}]\!](u, i)$$

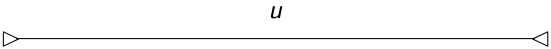
Nested weighted Automata vs wFO



We aim now at a logical characterization of w-Nested-Automata.

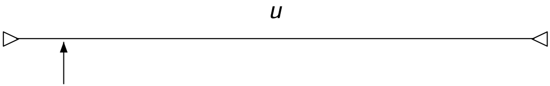
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



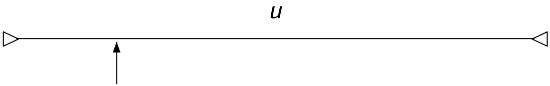
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



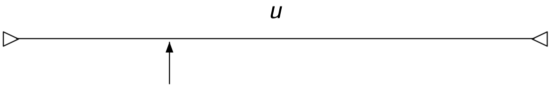
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



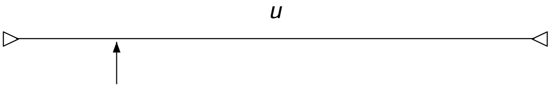
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



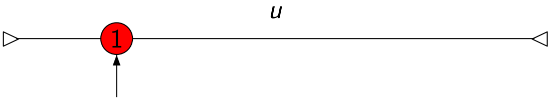
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



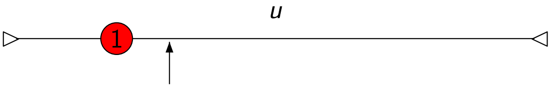
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



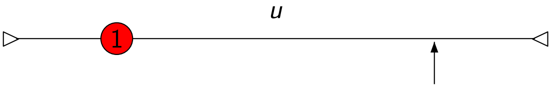
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



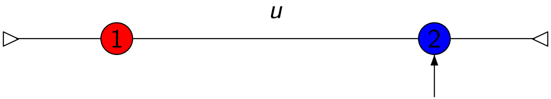
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



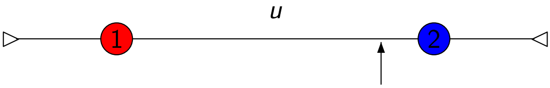
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



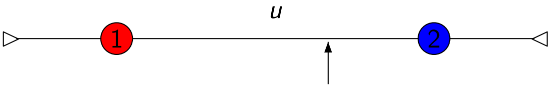
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



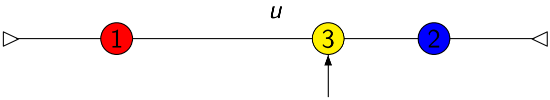
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



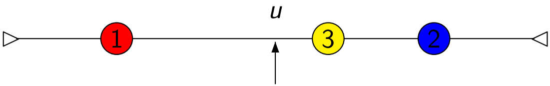
(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.

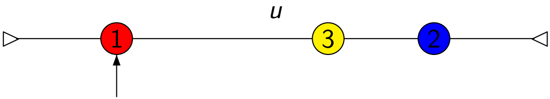


- Applicable transitions depend on current (state,letter,pebbles).

$$(p, ka, \text{Pebbles}, \text{D}, q), \text{ where D} \in \{\leftarrow, \rightarrow, \text{lift}, \text{drop}\}.$$

(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



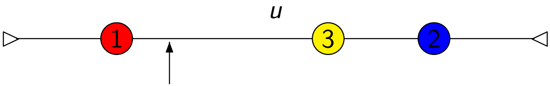
- Applicable transitions depend on current (state,letter,pebbles).

$$(p, ka, \text{Pebbles}, \text{D}, q), \text{ where D} \in \{\leftarrow, \rightarrow, \text{lift}, \text{drop}\}.$$

- Stack policy: only the most recently dropped pebble may be lifted

(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



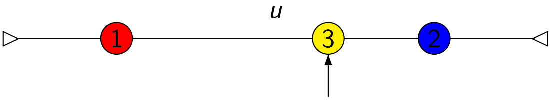
- Applicable transitions depend on current (state,letter,pebbles).

$$(p, ka, \text{Pebbles}, \text{D}, q), \text{ where D} \in \{\leftarrow, \rightarrow, \text{lift}, \text{drop}\}.$$

- Stack policy: only the most recently dropped pebble may be lifted
- Weak policy: pebble may be lifted only when the head scans its position.

(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



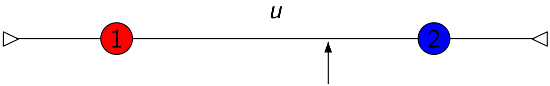
- Applicable transitions depend on current (state,letter,pebbles).

$$(p, ka, \text{Pebbles}, \text{D}, q), \text{ where D} \in \{\leftarrow, \rightarrow, \text{lift}, \text{drop}\}.$$

- Stack policy: only the most recently dropped pebble may be lifted
- Weak policy: pebble may be lifted only when the head scans its position.

(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



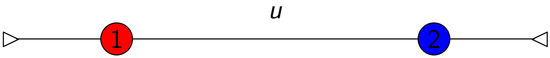
- Applicable transitions depend on current (state,letter,pebbles).

$$(p, ka, \text{Pebbles}, \text{D}, q), \text{ where } \text{D} \in \{\leftarrow, \rightarrow, \text{lift}, \text{drop}\}.$$

- Stack policy: only the most recently dropped pebble may be lifted
- Weak policy: pebble may be lifted only when the head scans its position.

(2-way) Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.



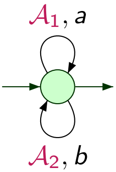
- Applicable transitions depend on current (state,letter,pebbles).

$$(p, ka, \text{Pebbles}, \text{D}, q), \text{ where D} \in \{\leftarrow, \rightarrow, \text{lift}, \text{drop}\}.$$

- Stack policy: only the most recently dropped pebble may be lifted
- Weak policy: pebble may be lifted only when the head scans its position.
- Note. For Boolean word automata, this does not add expressive power.

wPA can simulate wNA

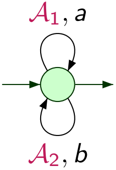
Proof by example: Consider the 1wNA



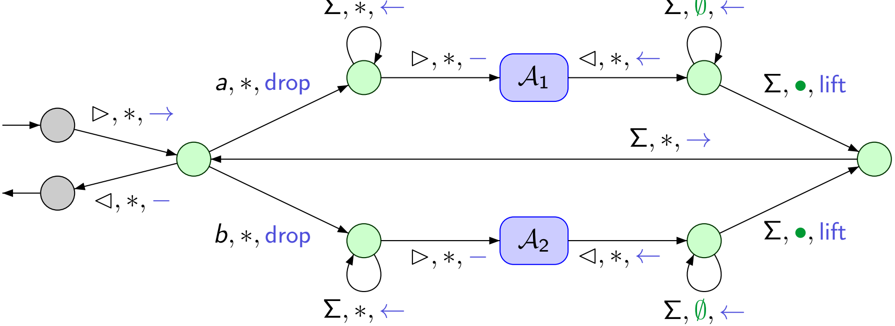
$$[\![\mathcal{A}]\!](u) = 3^{|u||u|_b} \cdot 2^{\sum \mathrm{pos}(a,u)}$$

wPA can simulate wNA

Proof by example: Consider the 1wNA



$$[\![\mathcal{A}]\!](u) = 3^{|u||u|_b} \cdot 2^{\sum \mathrm{pos}(a,u)}$$

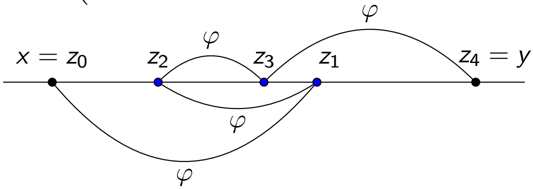


Transitive closure logics: TC and BTC

- For $\varphi(x, y)$ with (at least) two first order free variables, define

$$\varphi^1(x, y) = \varphi(x, y)$$

$$\varphi^n(x, y) = \exists z_0 \cdots \exists z_n \Big(x = z_0 \wedge z_n = y \wedge \mathrm{diff}(z_0, \ldots, z_n) \wedge \big[\textstyle\bigwedge_{1 \le \ell \le n} \varphi(z_{\ell-1}, z_\ell) \big] \Big).$$

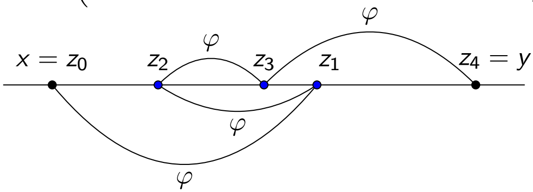


Transitive closure logics: TC and BTC

- For $\varphi(x, y)$ with (at least) two first order free variables, define

$$\varphi^1(x, y) = \varphi(x, y)$$

$$\varphi^n(x, y) = \exists z_0 \cdots \exists z_n \Big(x = z_0 \wedge z_n = y \wedge \mathrm{diff}(z_0, \ldots, z_n) \wedge \big[\bigwedge_{1 \le \ell \le n} \varphi(z_{\ell-1}, z_\ell) \big] \Big).$$



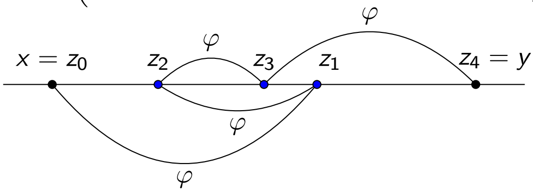
- The transitive closure operator is defined by $\mathrm{TC}_{xy}\varphi = \bigvee_{n \ge 1} \varphi^n$.

Transitive closure logics: TC and BTC

- For $\varphi(x, y)$ with (at least) two first order free variables, define

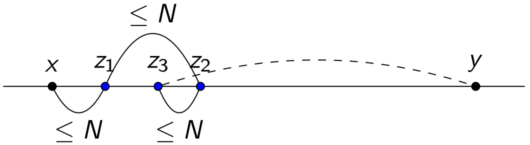
$$\varphi^1(x, y) = \varphi(x, y)$$

$$\varphi^n(x, y) = \exists z_0 \cdots \exists z_n \Big(x = z_0 \wedge z_n = y \wedge \mathrm{diff}(z_0, \ldots, z_n) \wedge \big[\bigwedge_{1 \leq \ell \leq n} \varphi(z_{\ell-1}, z_\ell)\big]\Big).$$



- The transitive closure operator is defined by $\mathrm{TC}_{xy}\varphi = \bigvee_{n \geq 1} \varphi^n$.

- Bounded transitive closure : $N\text{-}\mathrm{TC}_{xy}\varphi = \mathrm{TC}_{xy}(\varphi \wedge |x - y| \leq N)$



Bounded transitive closure and pebble automata

Express $N\text{-TC}_{xy}\varphi$ with 2 additional pebbles:

Given p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word (u, i, j)

Bounded transitive closure and pebble automata

Express N-$\mathrm{TC}_{xy}\varphi$ with 2 additional pebbles:

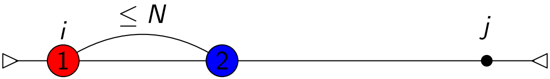
Given p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word (u, i, j)

1. $\mathcal{B}$ goes to i and drops pebble 1

Bounded transitive closure and pebble automata

Express N-$\mathrm{TC}_{xy}\varphi$ with 2 additional pebbles:

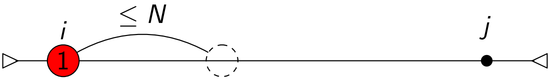
Given p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word (u, i, j)



1. $\mathcal{B}$ goes to i and drops pebble 1
2. $\mathcal{B}$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $\mathcal{B}$ simulates $\mathcal{A}$ on w where x and y are mapped to the positions of pebbles

Bounded transitive closure and pebble automata

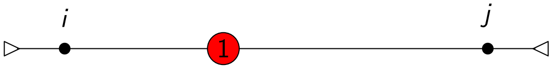
Express N-$\mathrm{TC}_{xy}\varphi$ with 2 additional pebbles:
Given p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word (u, i, j)



1. $\mathcal{B}$ goes to i and drops pebble 1
2. $\mathcal{B}$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $\mathcal{B}$ simulates $\mathcal{A}$ on w where x and y are mapped to the positions of pebbles
4. $\mathcal{B}$ lifts pebble 2 and pebble 1, and drops again pebble 1 where pebble 2 was.

Bounded transitive closure and pebble automata

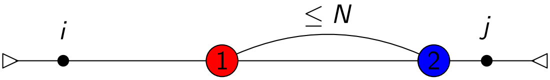
Express N-$\mathrm{TC}_{xy}\varphi$ with 2 additional pebbles:
Given p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word (u, i, j)



1. $\mathcal{B}$ goes to i and drops pebble 1
2. $\mathcal{B}$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $\mathcal{B}$ simulates $\mathcal{A}$ on w where x and y are mapped to the positions of pebbles
4. $\mathcal{B}$ lifts pebble 2 and pebble 1, and drops again pebble 1 where pebble 2 was.
5. If pebble 1 is not on j then goto 2 else stop.

Bounded transitive closure and pebble automata

Express N-$\mathrm{TC}_{xy}\varphi$ with 2 additional pebbles:
Given p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word (u, i, j)



1. $\mathcal{B}$ goes to i and drops pebble 1
2. $\mathcal{B}$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $\mathcal{B}$ simulates $\mathcal{A}$ on w where x and y are mapped to the positions of pebbles
4. $\mathcal{B}$ lifts pebble 2 and pebble 1, and drops again pebble 1 where pebble 2 was.
5. If pebble 1 is not on j then goto 2 else stop.

Bounded transitive closure and pebble automata

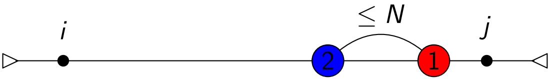
Express N-$\mathrm{TC}_{xy}\varphi$ with 2 additional pebbles:
Given p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word (u, i, j)

1. $\mathcal{B}$ goes to i and drops pebble 1
2. $\mathcal{B}$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $\mathcal{B}$ simulates $\mathcal{A}$ on w where x and y are mapped to the positions of pebbles
4. $\mathcal{B}$ lifts pebble 2 and pebble 1, and drops again pebble 1 where pebble 2 was.
5. If pebble 1 is not on j then goto 2 else stop.

Bounded transitive closure and pebble automata

Express N-$\mathrm{TC}_{xy}\varphi$ with 2 additional pebbles:

Given p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word (u, i, j)



1. $\mathcal{B}$ goes to i and drops pebble 1
2. $\mathcal{B}$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $\mathcal{B}$ simulates $\mathcal{A}$ on w where x and y are mapped to the positions of pebbles
4. $\mathcal{B}$ lifts pebble 2 and pebble 1, and drops again pebble 1 where pebble 2 was.
5. If pebble 1 is not on j then goto 2 else stop.

Bounded transitive closure and pebble automata

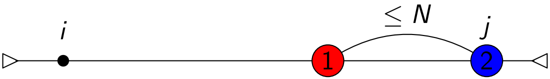
Express N-$\mathrm{TC}_{xy}\varphi$ with 2 additional pebbles:
Given p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word (u, i, j)

1. $\mathcal{B}$ goes to i and drops pebble 1
2. $\mathcal{B}$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $\mathcal{B}$ simulates $\mathcal{A}$ on w where x and y are mapped to the positions of pebbles
4. $\mathcal{B}$ lifts pebble 2 and pebble 1, and drops again pebble 1 where pebble 2 was.
5. If pebble 1 is not on j then goto 2 else stop.

Bounded transitive closure and pebble automata

Express $N\text{-TC}_{xy}\varphi$ with 2 additional pebbles:

Given p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word (u, i, j)

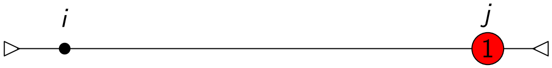


1. $\mathcal{B}$ goes to i and drops pebble 1
2. $\mathcal{B}$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $\mathcal{B}$ simulates $\mathcal{A}$ on w where x and y are mapped to the positions of pebbles
4. $\mathcal{B}$ lifts pebble 2 and pebble 1, and drops again pebble 1 where pebble 2 was.
5. If pebble 1 is not on j then goto 2 else stop.

Bounded transitive closure and pebble automata

Express N-$\mathrm{TC}_{xy}\varphi$ with 2 additional pebbles:

Given p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word (u, i, j)



1. $\mathcal{B}$ goes to i and drops pebble 1
2. $\mathcal{B}$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $\mathcal{B}$ simulates $\mathcal{A}$ on w where x and y are mapped to the positions of pebbles
4. $\mathcal{B}$ lifts pebble 2 and pebble 1, and drops again pebble 1 where pebble 2 was.
5. If pebble 1 is not on j then goto 2 else stop.

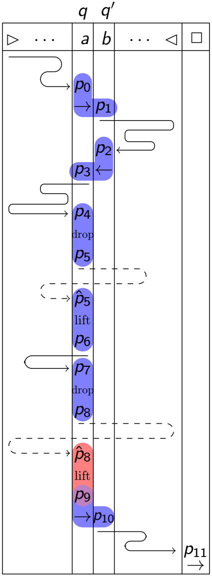
Expressiveness

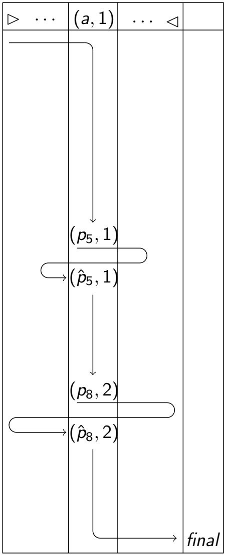
Theorem (Bollig, Gastin, Monmege, Zeitoun)

$$\mathrm{w(FO + BTC) = wPA = wNA}$$

- Proof of $\mathrm{w(FO + BTC)} \subseteq \mathrm{wPA}$ done in the previous slides
- Proof of $\mathrm{wPA} \subseteq \mathrm{wNA}$:
  Generalization of the translation of 2-way automata to 1-way automata.
- Proof of $\mathrm{wNA} \subseteq \mathrm{w(FO + BTC)}$:
  Generalization of a proof showing that weighted automata are expressible with transitive closure.

Flavor of the proof of 1-pebble ⊆ 1-nested

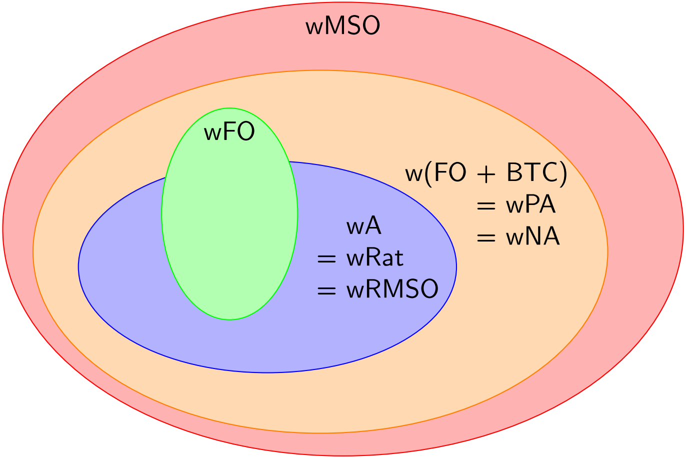




Requires commutativity

Summary

- Pebbles and nesting add expressive power in weighted automata.
- 2-way wA = 0-pebble wA = 0-nested wA = 1-way wA
- SAT of w(FO + BTC) is decidable for positive semiring



Open problems

Some closely related questions:

1. Unbounded steps in transitive closure?
2. Weak pebbles vs. strong pebbles?
3. Extended wRat for wPA?
4. Algorithms on wPA or wNA?

Open problems

Some closely related questions:

1. Unbounded steps in transitive closure?
2. Weak pebbles vs. strong pebbles?
3. Extended wRat for wPA?
4. Algorithms on wPA or wNA?

Extensions to other structures: Trees (ranked or unranked)

- Tree walking automata (TWA) are 2-way automata
- 1-way TWA = Depth First Search Automata (DFSA)
- Main Theorem (almost): w-Nested-DFSA = w(FO + $BTC^{<}$)

Open problems

Some closely related questions:

1. Unbounded steps in transitive closure?
2. Weak pebbles vs. strong pebbles?
3. Extended wRat for wPA?
4. Algorithms on wPA or wNA?

Extensions to other structures: Trees (ranked or unranked)

- Tree walking automata (TWA) are 2-way automata
- 1-way TWA = Depth First Search Automata (DFSA)
- Main Theorem (almost): w-Nested-DFSA = w(FO + $BTC^{<}$)
- pebble TWA $\stackrel{?}{=}$ nested DFSA

Open problems

Some closely related questions:

1. Unbounded steps in transitive closure?
2. Weak pebbles vs. strong pebbles?
3. Extended wRat for wPA?
4. Algorithms on wPA or wNA?

Extensions to other structures: Trees (ranked or unranked)

- Tree walking automata (TWA) are 2-way automata
- 1-way TWA = Depth First Search Automata (DFSA)
- Main Theorem (almost): w-Nested-DFSA = w(FO + $BTC^{<}$)
- pebble TWA $\stackrel{?}{=}$ nested DFSA
- Quantitative query languages: wXPath, wRXPath