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: $\mathbb{B} = (\{0, 1\}, \lor, \land, 0, 1)$

Automata $\mathcal{A}$

2-way pebbles alternating

Kleene

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

Elgot
Trakhtenbrot
Büchi

$E$

$\mathcal{V}$

eXtended Rational Expressions

FO + TC MSO sentences

eXtended FO + TC
Motivation: The Paradise for weights

Series $s : \Sigma^* \rightarrow K$

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

\[ w\text{FO} \subseteq w\text{A} = w\text{Rat} \subseteq w\text{MSO} \]
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 $w = a_1a_2 \cdots a_n$: product in the semiring.
  
  \[
  \text{weight}(p_0 \xrightarrow{k_1a_1} p_1 \xrightarrow{k_2a_2} \cdots \xrightarrow{k_na_n} p_n) = k_1k_2 \cdots k_n
  \]

- Value of a word: sum of all weights of runs on this word.
  
  \[
  [A](w) = \sum_{\rho \text{ run of } A \text{ on } w} \text{weight}(\rho) = \lambda \cdot \mu(w) \cdot \gamma
  \]

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

- $\mathbb{B} = (\{0, 1\}, \lor, \land, 0, 1)$
- $\mathbb{P} = (\mathbb{R}^+, +, \times, 0, 1)$
- $\mathbb{N} = (\mathbb{N}, +, \times, 0, 1)$
- $\mathbb{T} = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$
Examples of weighted automata

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

$$2\Sigma$$

$\mathcal{A}[u] = 2^{|u|}$ (deterministic)
Examples of weighted automata

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

  \[ [A](u) = 2^{|u|} \quad \text{(deterministic)} \]

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

  \[ [A](u) = |u|_a - |u|_b \]
Examples of weighted automata

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

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

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

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

  $\mathbb{Z}$, on $(\mathbb{Z}, +, \times, 0, 1)$

  $[A](u) = |u|_a - |u|_b$

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

  $\mathbb{N} \cup \{\infty\}$, on $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$

  $[A](ab^n c) = \min(3 + 2n, 6 + n)$
Weighted automata cannot compute large weights

Remark

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

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

- 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 $|u|$:

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

**Definition: Syntax of wMSO**

\[ \varphi ::= k \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \exists x \varphi \mid \forall x \varphi \mid \exists X \varphi \mid \forall X \varphi \]

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

**Definition: Semantics**

- A formula \( \varphi \) without free variables defines a mapping \([\varphi] : \Sigma^+ \rightarrow K\).
- 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**

\[ \varphi ::= k \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \exists x \, \varphi \mid \forall x \, \varphi \mid \exists X \, \varphi \mid \forall X \, \varphi \]

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

**Definition: Semantics**

- A formula \( \varphi \) without free variables defines a mapping \([\varphi] : \Sigma^+ \rightarrow K\).
- 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 \( y \)”.
- \([\varphi_1 \lor \varphi_2] = [\varphi_1] + [\varphi_2] \) and \([\varphi_1 \land \varphi_2] = [\varphi_1] \times [\varphi_2] \).

Remember: \( B = (\{0, 1\}, \lor, \land, 0, 1) \) and \( K = (K, +, \times, 0, 1) \).
Weighted MSO

Definition: Syntax of wMSO

\[ \varphi ::= k \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \exists x \varphi \mid \forall x \varphi \mid \exists X \varphi \mid \forall X \varphi \]

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

Definition: Semantics

- A formula \( \varphi \) without free variables defines a mapping \( \llbracket \varphi \rrbracket : \Sigma^+ \to K \).
- 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 \( y \)”.
- \( \llbracket \varphi_1 \lor \varphi_2 \rrbracket = \llbracket \varphi_1 \rrbracket + \llbracket \varphi_2 \rrbracket \) and \( \llbracket \varphi_1 \land \varphi_2 \rrbracket = \llbracket \varphi_1 \rrbracket \times \llbracket \varphi_2 \rrbracket \).
- \( \exists x \varphi \) interpreted as a sum over all positions.
- \( \forall x \varphi \) interpreted as a product over all positions.

Remember: \( B = (\{0, 1\}, \lor, \land, 0, 1) \) and \( K = (K, +, \times, 0, 1) \).
wMSO: examples

\[ \exists x P_a(x)(u) = \sum_{i \in \text{pos}(u)} [P_a(x)](u, i) = |u|_a \]
wMSO: examples

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

- $\forall y \ 2](u) = \prod_{i \in \mathrm{pos}(u)} [2](u, i) = 2^{\rvert u \rvert}$
wMSO: examples

\[ \exists x \ P_a(x) \] \( (u) = \sum_{i \in \text{pos}(u)} [P_a(x)](u, i) = |u|^a \)

\[ \forall y \ 2 \] \( (u) = \prod_{i \in \text{pos}(u)} [2](u, i) = 2^{|u|} \)

\[ \forall x \ \forall y \ 2 \] \( (u) = \prod_{i \in \text{pos}(u)} [\forall y \ 2](u, i) = (2^{|u|})^{|u|} = 2^{|u|^2} \).

\begin{itemize}
  \item \text{recognizable}
  \item \text{recognizable}
  \item not recognizable
\end{itemize}

\text{w-Automata are not closed under universal quantification.}
**wMSO: examples**

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

- \[ \forall y \ 2](u) = \prod_{i \in \text{pos}(u)} [2](u, i) = 2^{|u|} \]
  - recognizable

- \[ \forall x \forall y \ 2](u) = \prod_{i \in \text{pos}(u)} \forall y \ 2](u, i) = (2^{|u|})^{|u|} = 2^{|u|^2}. \]
  - 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 \( \lor \land \) 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.

\[
[A_1](u) = 2^{i+1} \text{ if } u \in \Sigma_0^i \Sigma_1 \Sigma_0^*
\]

\[
[A_2](u) = 3^{|u|}
\]
Nested automata (= 1-way pebble automata)

A 0-nested wA is a classical weighted automaton.

\[ \llbracket A_1 \rrbracket(u) = 2^{i+1} \text{ if } u \in \Sigma_0^i \Sigma_1 \Sigma_0^* \]

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

\( A_{p,a,q} \) restarts on \((u, i)\) and computes the weight \( p \xrightarrow{\llbracket A_{p,a,q} \rrbracket(u,i)} q \).

\[ \llbracket A \rrbracket(u) = 3 |u| |u|_b \cdot 2^{\sum_{pos(a,u)}} \]
Nested automata (\(=\) 1-way pebble automata)

A 0-nested wA is a classical weighted automaton.

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

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

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 \(p \xrightarrow{[\mathcal{A}_{p,a,q}](u,i)_{a}} q\).

An \(r\)-nested automaton does \(1 + |u| + |u|^2 + \cdots + |u|^r\) 1-way runs on a word \(u\).
Nested automata are closed under $\exists \forall$

Proof: $\forall x \ A(x)$

$$A, \Sigma$$

$$[B](u) = \prod_{i=1}^{\mid u \mid} [A](u, i)$$
Nested automata are closed under $\exists \forall$

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

![Diagram for $\forall x \; \mathcal{A}(x)$]

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

![Diagram for $\exists x \; \mathcal{A}(x)$]

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

\[
[\mathcal{B}](u) = \sum_{i=1}^{\left|u\right|} [\mathcal{A}](u, i)
\]
Nested weighted Automata vs wFO

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

wNA = ???

wA = wRat = wRMSO
(2-way) Pebble weighted automata

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

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

\[ u \]
(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\}.
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\}.

\[ u \]

Diagram: 1 \rightarrow 2 with pebble movement.
(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). 
  \((p, ka, \text{Pebbles, D, } q)\), where \(D \in \{\leftarrow, \rightarrow, \text{lift, drop}\}\).
(2-way) Pebble weighted automata

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

- Applicable transitions depend on current (state, letter, pebbles) $\langle p, ka, \text{Pebbles}, D, q \rangle$, where $D \in \{\leftarrow, \rightarrow, \text{lift, drop}\}$.

- 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). $(p, ka, \text{Pebbles}, D, q)$, 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.
Automaton with 2-way mechanism and pebbles \( \{1, \ldots, r\} \).

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

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

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

- 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 \((\text{state,letter,pebbles})\), where \(D \in \{ \leftarrow, \rightarrow, \text{lift, drop}\} \).

- 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

\[ [\mathcal{A}](u) = 3|u||u|_b \cdot 2\sum_{\text{pos}(a,u)} \]
wPA can simulate wNA

Proof by example: Consider the 1wNA

\[ \mathcal{A}_1, a \]

\[ \mathcal{A}_2, b \]

\[ [\mathcal{A}] (u) = 3 |u||u|_b \cdot 2 \sum \text{pos}(a, u) \]
For $\varphi(x, y)$ with (at least) two first order free variables, define

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

$$
\varphi^n(x, y) = \exists z_0 \cdots \exists z_n (x = z_0 \land z_n = y \land \text{diff}(z_0, \ldots, z_n) \land [\bigwedge_{1 \leq \ell \leq n} \varphi(z_{\ell-1}, z_\ell)]).
$$
Transitive closure logics: TC and BTC

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

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

  $\varphi^n(x, y) = \exists z_0 \cdots \exists z_n \left( x = z_0 \land z_n = y \land \text{diff}(z_0, \ldots, z_n) \land \left[ \bigwedge_{1 \leq \ell \leq n} \varphi(z_{\ell-1}, z_{\ell}) \right] \right)$. 

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

![Diagram](attachment:image.png)
Transitive closure logics: TC and BTC

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

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

$$\varphi^n(x, y) = \exists z_0 \cdots \exists z_n \left( x = z_0 \land z_n = y \land \text{diff}(z_0, \ldots, z_n) \land \left[ \bigwedge_{1 \leq \ell \leq n} \varphi(z_{\ell-1}, z_\ell) \right] \right).$$

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

- Bounded transitive closure: $N$-$\text{TC}_{xy} \varphi = \text{TC}_{xy} (\varphi \land |x - y| \leq N)$
Bounded transitive closure and pebble automata

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

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

![Diagram of i and j with additional pebbles]
Express $N\text{-}TC_{xy}\varphi$ with 2 additional pebbles:

Given $p$-pebble automaton $\mathcal{A}$ on $\Sigma_{xy}$ recognizing $\llbracket \varphi \rrbracket$ and a word $(u, i, j)$

1. $B$ goes to $i$ and drops pebble 1
Express $N\text{-TC}_{xy} \varphi$ with 2 additional pebbles:

Given $p$-pebble automaton $\mathcal{A}$ on $\Sigma_{xy}$ recognizing $\llbracket \varphi \rrbracket$ and a word $(u, i, j)$

1. $B$ goes to $i$ and drops pebble 1
2. $B$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $B$ simulates $\mathcal{A}$ on $w$ where $x$ and $y$ are mapped to the positions of pebbles
Express $N\text{-TC}_{xy}\varphi$ with 2 additional pebbles:
Given $p$-pebble automaton $\mathcal{A}$ on $\Sigma_{xy}$ recognizing $[\varphi]$ and a word $(u, i, j)$

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

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

1. $B$ goes to $i$ and drops pebble 1
2. $B$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $B$ simulates $\mathcal{A}$ on $w$ where $x$ and $y$ are mapped to the positions of pebbles
4. $B$ 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.
Express $N\text{-TC}_{xy} \varphi$ with 2 additional pebbles:
Given $p$-pebble automaton $\mathcal{A}$ on $\Sigma_{xy}$ recognizing $[\varphi]$ and a word $(u, i, j)$

1. $B$ goes to $i$ and drops pebble 1
2. $B$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $B$ simulates $\mathcal{A}$ on $w$ where $x$ and $y$ are mapped to the positions of pebbles
4. $B$ 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.
Express $N$-$\text{TC}_{xy}\varphi$ with 2 additional pebbles:

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

1. $B$ goes to $i$ and drops pebble 1
2. $B$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $B$ simulates $\mathcal{A}$ on $w$ where $x$ and $y$ are mapped to the positions of pebbles
4. $B$ 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.
Express $N$-TC$_{xy} \varphi$ with 2 additional pebbles:

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

1. $B$ goes to $i$ and drops pebble 1
2. $B$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $B$ simulates $A$ on $w$ where $x$ and $y$ are mapped to the positions of pebbles
4. $B$ 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.
Express $N\text{-TC}_{xy}\varphi$ with 2 additional pebbles:

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

1. $B$ goes to $i$ and drops pebble 1
2. $B$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $B$ simulates $A$ on $w$ where $x$ and $y$ are mapped to the positions of pebbles
4. $B$ 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.
Express $N$-TC$_{xy} \varphi$ with 2 additional pebbles:

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

1. $B$ goes to $i$ and drops pebble 1
2. $B$ drops nondeterministically pebble 2 on a position at distance $\leq N$
3. $B$ simulates $A$ on $w$ where $x$ and $y$ are mapped to the positions of pebbles
4. $B$ 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.
Express $N$-TC$_{xy} \varphi$ with 2 additional pebbles:

Given $p$-pebble automaton $\mathcal{A}$ on $\Sigma_{xy}$ 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)**

\[ w(\text{FO} + \text{BTC}) = w\text{PA} = w\text{NA} \]

- Proof of \( w(\text{FO} + \text{BTC}) \subseteq w\text{PA} \) done in the previous slides
- Proof of \( w\text{PA} \subseteq w\text{NA} \): Generalization of the translation of 2-way automata to 1-way automata.
- Proof of \( w\text{NA} \subseteq w(\text{FO} + \text{BTC}) \): Generalization of a proof showing that weighted automata are expressible with transitive closure.
Flavor of the proof of $1$-pebble $\subseteq 1$-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 $w(\text{FO} + \text{BTC})$ is decidable for positive semiring $wMSO$

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

$wA = w\text{Rat} = wRMSO$
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): \( \text{w-Nested-DFSA} = \text{w}(\text{FO} + \text{BTC}^<) \)
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 = $w(\text{FO} + \text{BTC}^<)$
▶ pebble TWA $\not\equiv$ 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): \( w\text{-Nested-DFSA} = w(FO + BTC^<) \)
- Pebble TWA \( \equiv \) nested DFSA
- Quantitative query languages: wXPath, wRXPath