Pebble weighted automata and transitive closure logics

Benjamin Monmege

Benedikt Bollig, Paul Gastin, Marc Zeitoun
LSV, ENS Cachan, CNRS, INRIA.

ICALP 2010

July 10, 2010

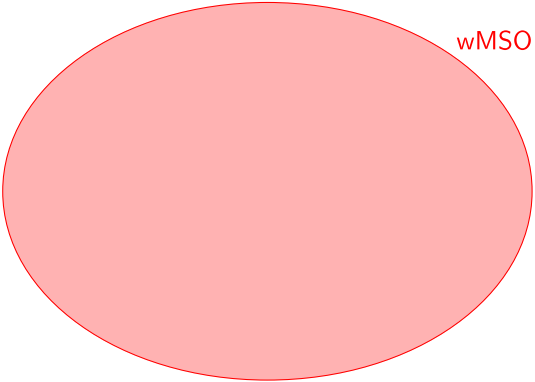
Motivations

Sequential setting: automata on (finite) words.

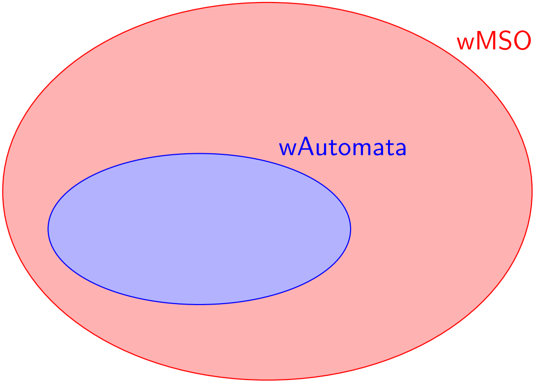
- Weighted automata: quantitative extension of classical automata.
  - Classical: decide whether a given word is accepted or not,
  - Weighted: compute a value in a semiring from input word.
- Weighted logics: a formula evaluated on a word produces a value.
  - How often does a Boolean property hold?
  - Is the number of nodes selected by a request at least 10?
- In this talk, we focus on expressiveness. Boolean setting:

$$\text{Automata} = \text{FO+TC} = \text{MSO} = \text{EMSO} = \dots$$

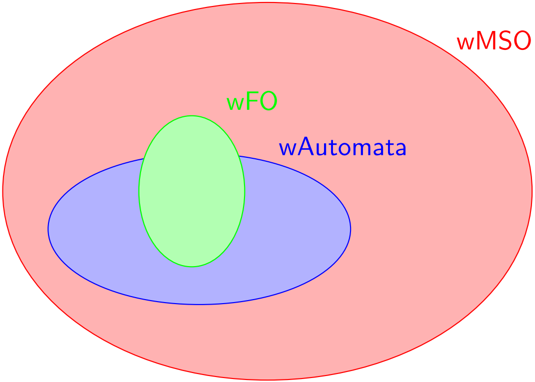
Expressivity in weighted setting



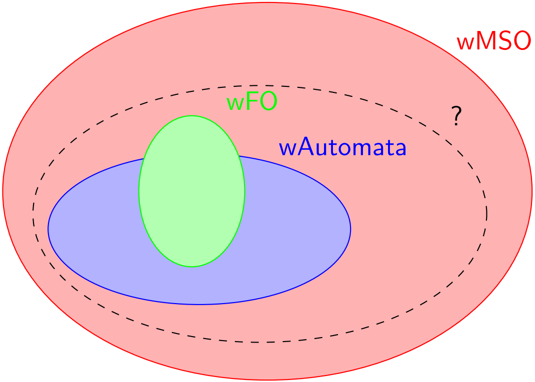
Expressivity in weighted setting



Expressivity in weighted setting



Expressivity in weighted setting



Weighted automata

- Transitions carry weights from a semiring $\mathbb{K} = (K, +, \times, 0, 1)$.

Weighted automata

- Transitions carry weights from a semiring $\mathbb{K} = (K, +, \times, 0, 1)$.

- Weight of a run: product of all transition weights in the semiring.

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

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

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

Weighted automata

- Transitions carry weights from a semiring $\mathbb{K} = (K, +, \times, 0, 1)$.

- Weight of a run: product of all transition weights in the semiring.

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

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

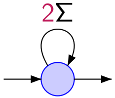
$$[\![\mathcal{A}]\!](w) = \sum_{\rho \text{ run of } \mathcal{A} \text{ on } w} \text{weight}(\rho)$$

Example: Semirings

- $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$ — 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.

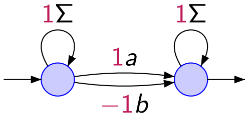
Examples of weighted automata

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



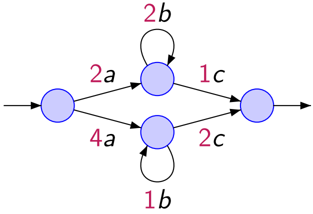
$[\![\mathcal{A}]\!](u) = 2^{|u|}$ (deterministic)

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



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

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



$$[\![\mathcal{A}]\!](ab^n c) = \begin{cases} 3 + 2n & \text{if } n \leq 3 \\ 6 + n & \text{if } n \geq 4 \end{cases}$$

Weighted automata cannot compute large weights

Lemma

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

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

- There are $O(|Q|^{|u|})$ runs on u,
- Each of which of weight exponential in $|u| = n$: $k_1 \cdots k_n \leq (\max k_i)^n$.

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 $[\![\varphi_1 \wedge \varphi_2]\!] = [\![\varphi_1]\!] \times [\![\varphi_2]\!]$.
- $\exists x\, \varphi$ interpreted as a sum over all positions.
- $\forall x\, \varphi$ interpreted as a product over all positions.

wMSO: examples

- $[\![\exists x\ P_a(x)]\!] = |u|_a$ recognizable
- $[\![\exists x\ P_a(x)\ \vee\ \exists x\ [-1 \wedge P_b(x)]]\!] = |u|_a - |u|_b$ recognizable
- $[\![\forall y\ 2]\!](u) = 2^{|u|}$. recognizable

wMSO: examples

- $[\![\exists x\ P_a(x)]\!] = |u|_a$ recognizable
- $[\![\exists x\ P_a(x)\ \vee\ \exists x\ [-1 \wedge P_b(x)]]\!] = |u|_a - |u|_b$ recognizable
- $[\![\forall y\ 2]\!](u) = 2^{|u|}$. recognizable
- $[\![\forall x \forall y\ 2]\!](u) = 2^{|u|^2}$. not recognizable
- $\Longrightarrow$ Recognizable are not stable under universal quantification.

wMSO: examples

- $[\![\exists x\ P_a(x)]\!] = |u|_a$ recognizable
- $[\![\exists x\ P_a(x)\ \vee\ \exists x\ [-1 \wedge P_b(x)]]\!] = |u|_a - |u|_b$ recognizable
- $[\![\forall y\ 2]\!](u) = 2^{|u|}$. recognizable
- $[\![\forall x\, \forall y\ 2]\!](u) = 2^{|u|^2}$. not recognizable
- $\Longrightarrow$ Recognizable are not stable under universal quantification.

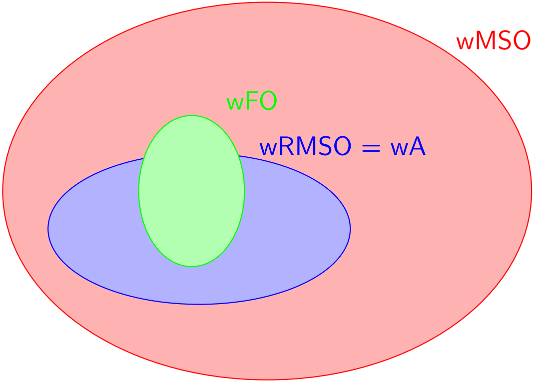
[DG'05] defined wRMSO, a fragment of wMSO (no second order universal quantifications, and first order universal quantifications restricted over *simple formulas*)

Theorem (Droste & Gastin'05)

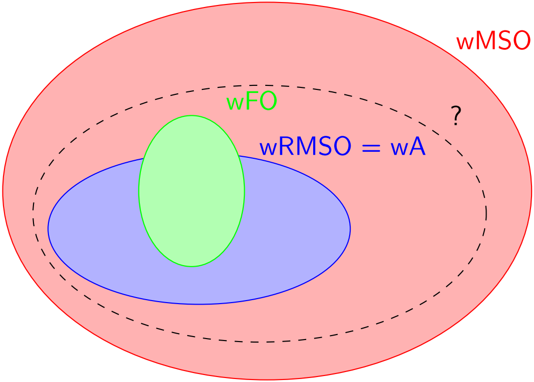
Weighted automata = wRMSO

(effective translation).

Another way of thinking ?



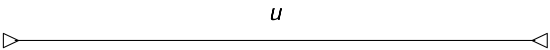
Another way of thinking ?



- Extension of weighted automata to obtain a bigger class of power series : in particular closed by first order quantifications
- Express the new model in an extension of weighted first order logic

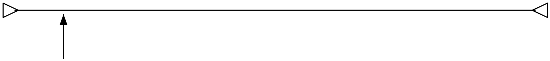
Pebble weighted automata

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



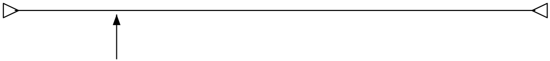
Pebble weighted automata

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



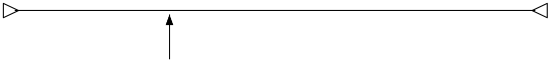
Pebble weighted automata

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



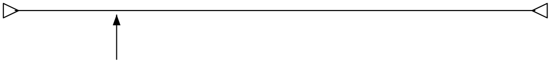
Pebble weighted automata

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



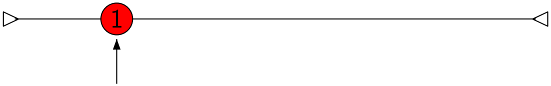
Pebble weighted automata

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



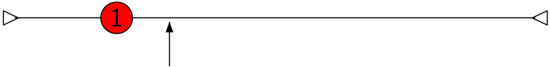
Pebble weighted automata

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



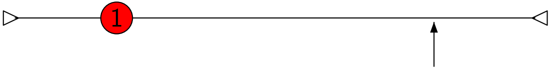
Pebble weighted automata

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



Pebble weighted automata

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

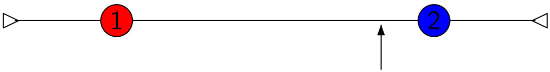


Pebble weighted automata

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

Pebble weighted automata

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

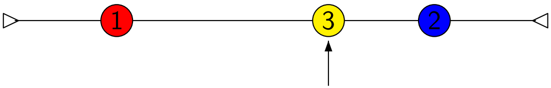


Pebble weighted automata

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

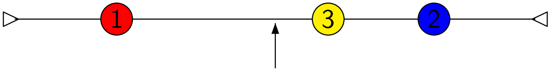
Pebble weighted automata

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



Pebble weighted automata

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

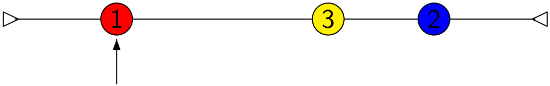


- Applicable transitions depend on current (state,letter,pebbles).

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

Pebble weighted automata

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



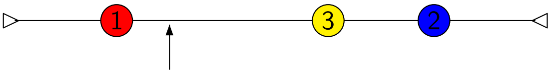
- Applicable transitions depend on current (state,letter,pebbles).

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

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

Pebble weighted automata

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



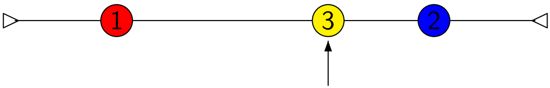
- Applicable transitions depend on current (state,letter,pebbles).

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

- Stack policy : only the most recently dropped pebble may be lifted
- Weak pebbles : liftable only at the position of the pebble

Pebble weighted automata

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



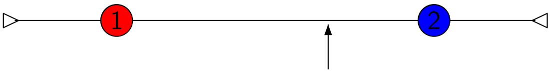
- Applicable transitions depend on current (state,letter,pebbles).

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

- Stack policy : only the most recently dropped pebble may be lifted
- Weak pebbles : liftable only at the position of the pebble

Pebble weighted automata

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



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

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

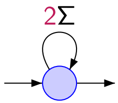
- Stack policy : only the most recently dropped pebble may be lifted
- Weak pebbles : liftable only at the position of the pebble

Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1, \ldots, r\}$.
- Applicable transitions depend on current (state,letter,pebbles).

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

- Stack policy : only the most recently dropped pebble may be lifted
- Weak pebbles : liftable only at the position of the pebble
- Note. For Boolean word automata, this does not add expressive power.

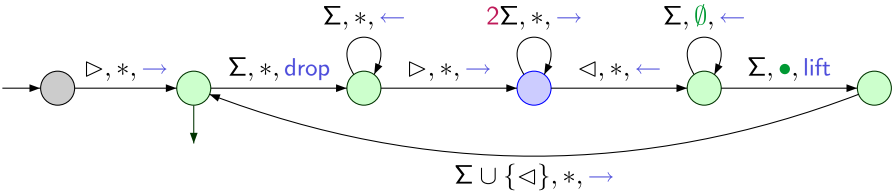


Pebble weighted automata

- Automaton with 2-way mechanism and pebbles $\{1,\ldots,r\}$.
- Applicable transitions depend on current (state,letter,pebbles).

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

- Stack policy : only the most recently dropped pebble may be lifted
- Weak pebbles : liftable only at the position of the pebble

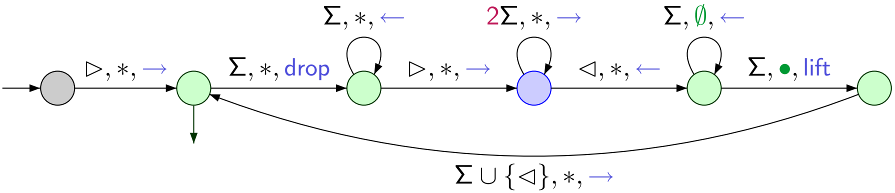


Pebble weighted automata

- 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 pebbles : liftable only at the position of the pebble



- Computes $2^{|u|^2}$: pebbles add expressive power.

Pebble weighted automata are stable under wFO

Definition: Weighted First-order logic

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

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

Pebble weighted automata are stable under wFO

Lemma

Pebble weighted automata are stable under wFO constructs.

Proof idea for $\forall$: consider a p-pebble automaton $\mathcal{A}$ over Σ_x, we want to compute $\forall x\ \mathcal{A}(x)$. Add first pebble interpreted as free variable.

Drop pebble 1 successively on each position.

$\widetilde{\mathcal{A}}$

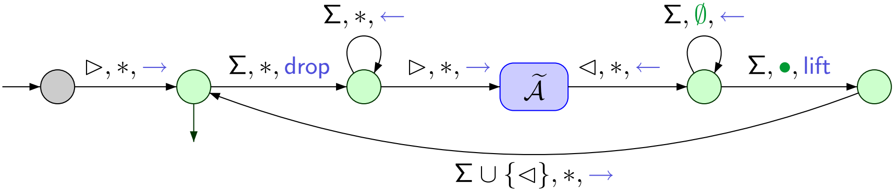
Pebble weighted automata are stable under wFO

Lemma

Pebble weighted automata are stable under wFO constructs.

Proof idea for ∀: consider a p-pebble automaton $\mathcal{A}$ over Σ_x, we want to compute $\forall x\, \mathcal{A}(x)$. Add first pebble interpreted as free variable.

Drop pebble 1 successively on each position.



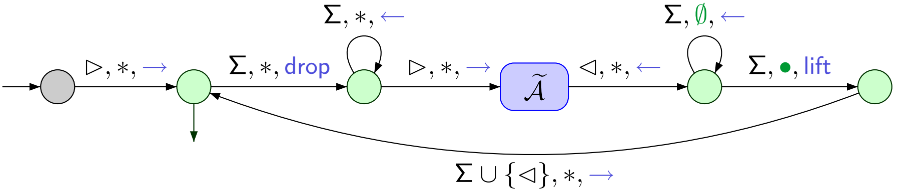
Pebble weighted automata are stable under wFO

Lemma

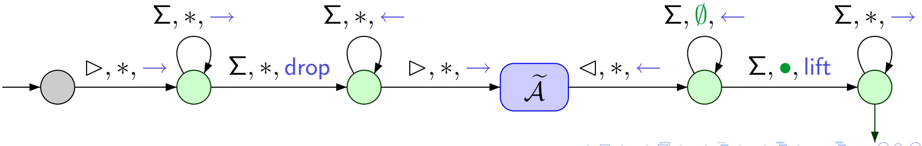
Pebble weighted automata are stable under wFO constructs.

Proof idea for $\forall$: consider a p-pebble automaton $\mathcal{A}$ over Σ_x, we want to compute $\forall x\, \mathcal{A}(x)$. Add first pebble interpreted as free variable.

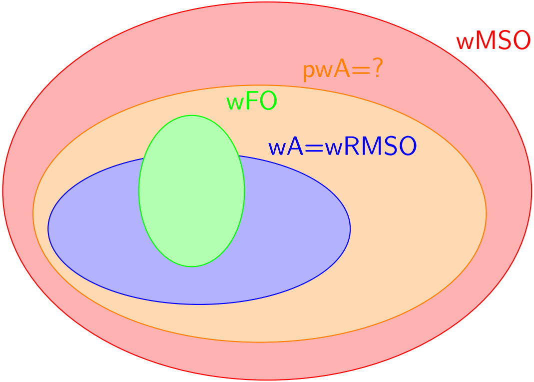
Drop pebble 1 successively on each position.



For $\exists$: nondeterministically drop pebble 1.



Pebble weighted Automata vs. wFO

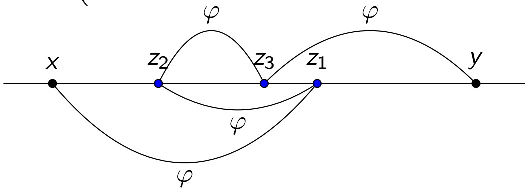


Transitive closure logics

- For φ with at least two first order free variables, define

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

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

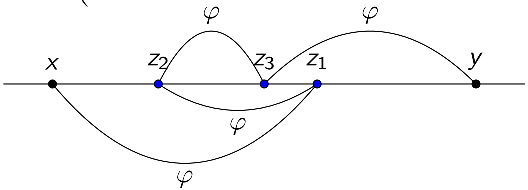


Transitive closure logics

- For φ with at least two first order free variables, define

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

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



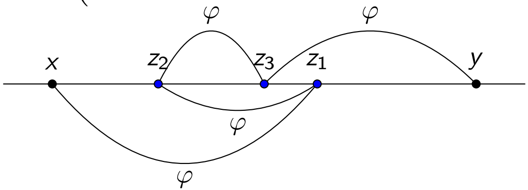
- The transitive closure operator is defined by $\mathrm{TC}_{xy}\varphi = \bigvee_{n \ge 1} \varphi^n$.

Transitive closure logics

- For φ with at least two first order free variables, define

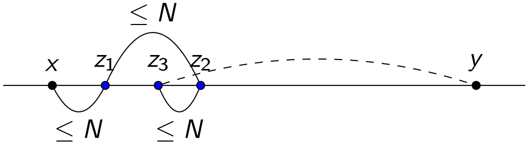
$$\varphi^1(x,y) = \varphi(x,y)$$

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



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

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



Bounded transitive closure and pebble automata

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

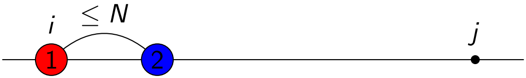
Bounded transitive closure and pebble automata

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

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

Bounded transitive closure and pebble automata

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



1. $\mathcal{B}$ goes to i and drops pebble 1
2. Choose nondeterministically a position at distance $\leq N$ and drops pebble 2
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: p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word $(w, x \to i, y \to j)$

1. $\mathcal{B}$ goes to i and drops pebble 1
2. Choose nondeterministically a position at distance $\leq N$ and drops pebble 2
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 drop again pebble 1.

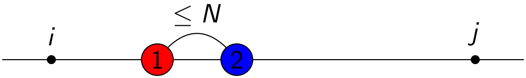
Bounded transitive closure and pebble automata

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

1. $\mathcal{B}$ goes to i and drops pebble 1
2. Choose nondeterministically a position at distance $\leq N$ and drops pebble 2
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 drop again pebble 1.

Bounded transitive closure and pebble automata

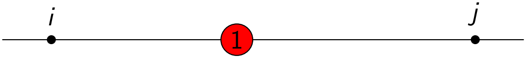
- Express N-$\mathrm{TC}_{xy}\varphi$ with 2 additional pebbles: p-pebble automaton $\mathcal{A}$ on Σ_{xy} recognizing $[\![\varphi]\!]$ and a word $(w, x \to i, y \to j)$



1. $\mathcal{B}$ goes to i and drops pebble 1
2. Choose nondeterministically a position at distance $\leq N$ and drops pebble 2
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 drop again pebble 1.

Bounded transitive closure and pebble automata

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



1. $\mathcal{B}$ goes to i and drops pebble 1
2. Choose nondeterministically a position at distance $\leq N$ and drops pebble 2
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 drop again pebble 1.

- Question: how to extend this result to unbounded steps ?

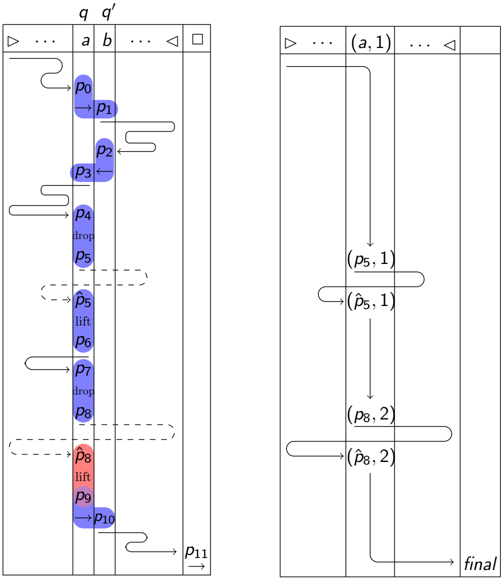
Expressiveness

Theorem (Bollig, Gastin, M., Zeitoun)

wFO + TC with bounded steps = weighted pebble automata.

- Proof of $\subseteq$ done by the previous slides
- Proof of $\supseteq$ generalizes the translation 2-way $\to$ 1-way automata.
- Uses an intermediate automaton model (nested automata): one-way automata than can do several runs by marking some positions to keep informations

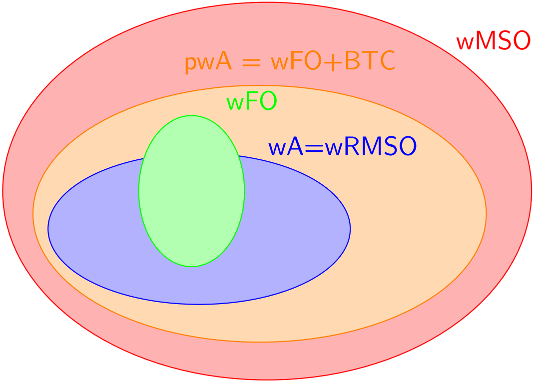
Flavor of the proof of $\supseteq$: 1 pebble $\rightarrow$ 1 nested



Summary and some short-term questions

Summary:

- Pebbles add expressive power in weighted extensions of automata
- Natural logical equivalence of pebble jumps and transitive closure steps



Summary and some short-term questions

Summary:

- Pebbles add expressive power in weighted extensions of automata
- Natural logical equivalence of pebble jumps and transitive closure steps

Perspectives:

1. Algorithms. (SAT is decidable for positive semiring.)
2. Relax bounded assumption.
3. Weak pebbles vs. strong pebbles.
4. Extension of weighted pebbles automata to others structures : trees (and query language XPath...), infinite words