

# Weighted expressions and one-way pebble automata

Marc Zeitoun

Benedikt Bollig, Paul Gastin, Benjamin Monmege  
LaBRI, U. Bordeaux, LSV, ENS Cachan, CNRS, INRIA

LIAFA  
May 20, 2011

Preliminary version at ICALP'10

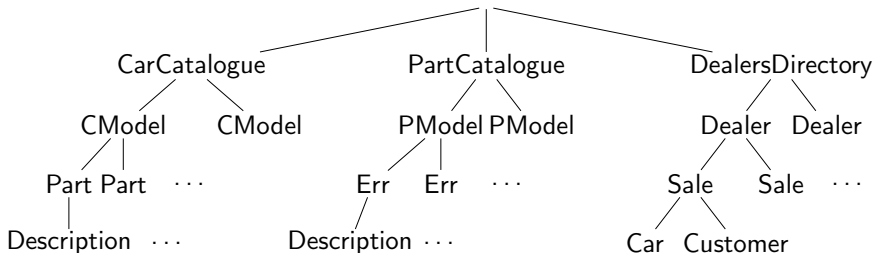
# Motivation: specifying quantitative properties

Develop high-level denotational formalism

- ▶ to express **quantitative** properties on words/trees,
- ▶ should be **flexible**,
- ▶ should allow us to compute **arithmetic expressions** (possibly **guarded by logical conditions** written in a standard language (eg., FO or XPath),
- ▶ should have an **equivalent operational model**.

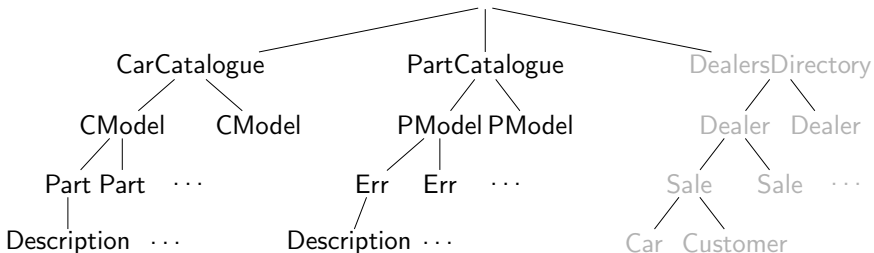
## Motivation: an example

- ▶ An XML document for car models, pieces, and dealers.



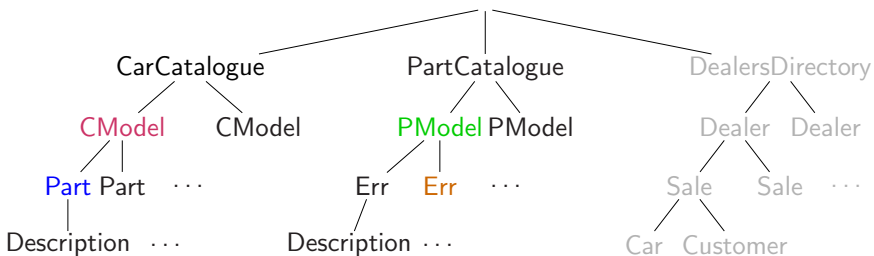
## Motivation: an example

- ▶ An XML document for car models, pieces, and dealers.



## Motivation: an example

- ▶ An XML document for car models, pieces, and dealers.

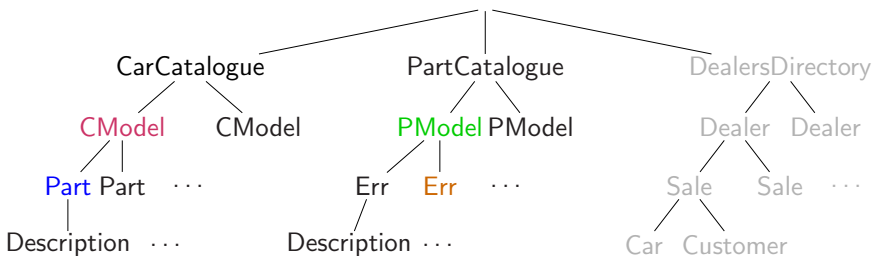


- ▶ *There is a car model using some car part with an error.*  $\exists xyz t \varphi(x, y, z, t)$

$$\varphi = [\text{CModel}(x) \wedge \text{Part}(y) \wedge x <_v y] \wedge [\text{PModel}(z) \wedge \text{Err}(t) \wedge z <_v t] \wedge \text{Match}(y, z).$$

## Motivation: an example

- ▶ An XML document for car models, pieces, and dealers.



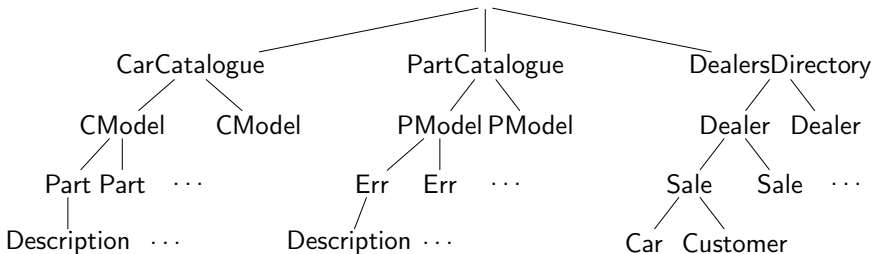
- ▶ *There is a car model using some car part with an error.*  $\exists xyz t \varphi(x, y, z, t)$

$$\varphi = [\text{CModel}(x) \wedge \text{Part}(y) \wedge x <_v y] \wedge [\text{PModel}(z) \wedge \text{Err}(t) \wedge z <_v t] \wedge \text{Match}(y, z).$$

- ▶ *Total number of errors:*  $\bigoplus_{x,y,z,t} \varphi(x, y, z, t)$ , and compute in  $(\mathbb{N}, +, \times)$ .

## Motivation: an example

- ▶ An XML document for car models, pieces, and dealers.

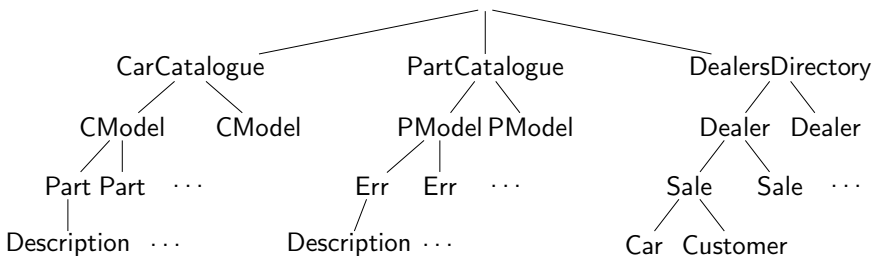


- ▶ XML document now includes car dealers and performed sales.
- ▶ *Maximal number of errors to be fixed per dealer:*

$$\text{Max}_d \text{ Dealer}(d) \wedge \sum_u \left[ d <_v u \wedge \text{Car}(u) \wedge \sum_{x,y,z,t} \text{Match}'(u,x) \wedge \varphi(x,y,z,t) \right]$$

## Motivation: an example

- ▶ An XML document for car models, pieces, and dealers.



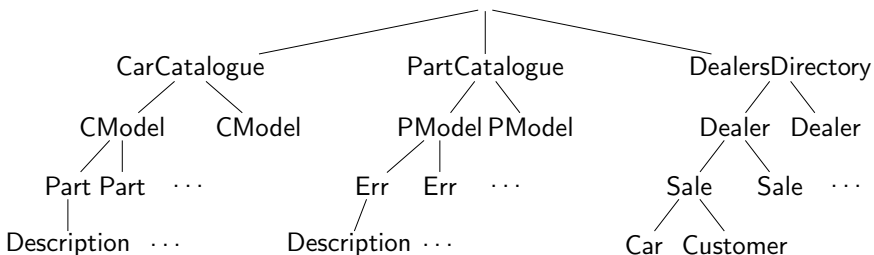
- ▶ XML document now includes car dealers and performed sales.
- ▶ *Maximal number of errors to be fixed per dealer:*

$$\text{Max}_d \sum_{u,x,y,z,t} \psi(d, u, x, y, z, t) \rightarrow 1$$



## Motivation: an example

- ▶ An XML document for car models, pieces, and dealers.

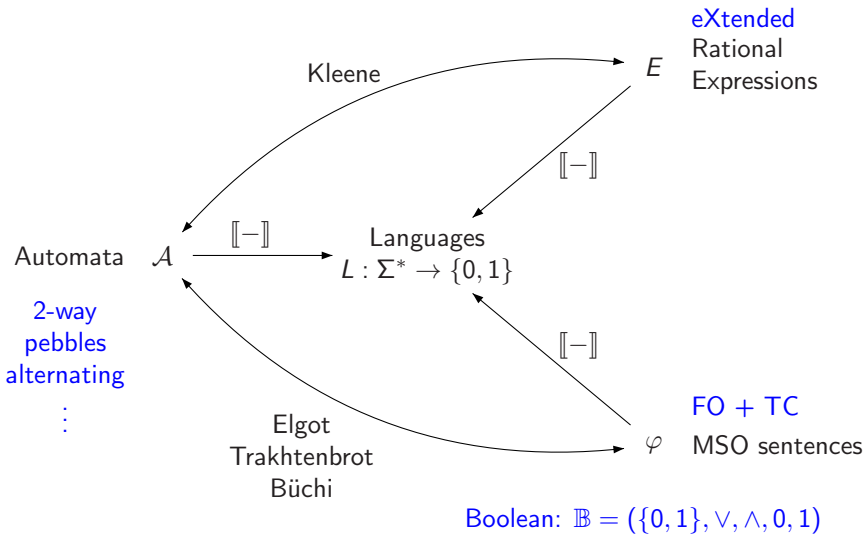


- ▶ XML document now includes car dealers and performed sales.
- ▶ *Maximal number of errors to be fixed per dealer:*

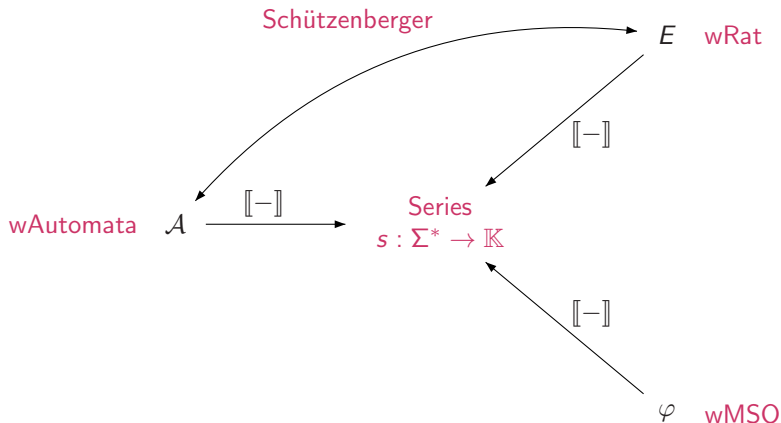
$$\otimes_d \oplus_{u,x,y,z,t} \psi(d, u, x, y, z, t) \rightarrow 1$$

- ▶ Same formalism: compute in the **max-plus** semiring  $(\mathbb{N} \cup \{-\infty\}, \max, +)$ .

# On words: what remains true for weights?

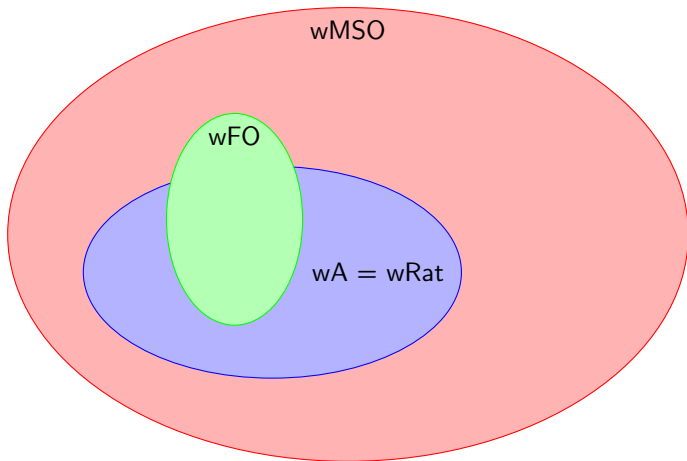


# On words: what remains true for weights?

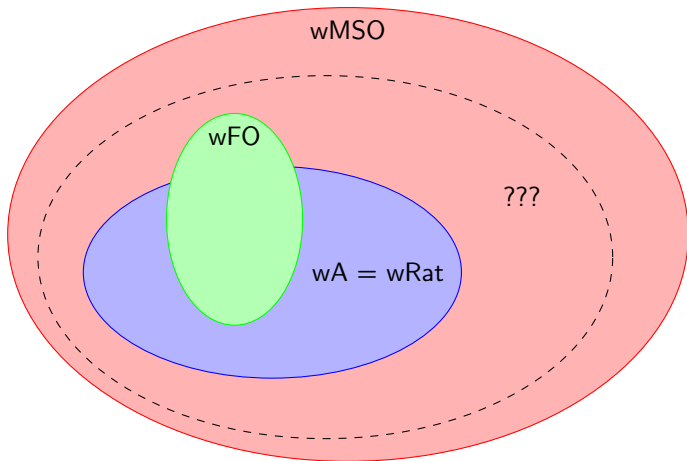


Quantitative:  $\mathbb{K} = (K, +, \times, 0, 1)$

# Expressiveness in the weighted setting



# Expressiveness in the weighted setting



Find a robust class containing both  $wFO$  and  $wAutomata$ .

# Weighted automata

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



- ▶ **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  $\mathbb{K}$ :  $\mu : \Sigma \rightarrow K^{Q \times Q}$ .



- ▶ **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      |

# Weighted automata

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



- ▶ **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.

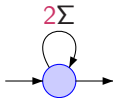
$$[[\mathcal{A}]](w) = \sum_{\rho \text{ run of } \mathcal{A} \text{ on } w} \text{weight}(\rho) = \lambda \cdot \mu(w) \cdot \gamma$$

- ▶ In this talk: semiring is **commutative**.



# Examples of weighted automata

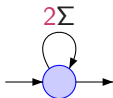
- ▶ Alphabet  $\Sigma$ , on  $(\mathbb{N}, +, \times, 0, 1)$



$$\llbracket \mathcal{A} \rrbracket(u) = 2^{|u|} \quad (\text{deterministic})$$

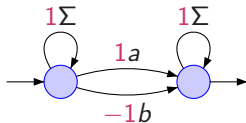
# Examples of weighted automata

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



$$\llbracket \mathcal{A} \rrbracket(u) = 2^{|u|} \quad (\text{deterministic})$$

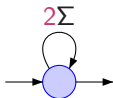
- ▶ Alphabet  $\Sigma = \{a, b\}$ , on  $(\mathbb{Z}, +, \times, 0, 1)$



$$\llbracket \mathcal{A} \rrbracket(u) = |u|_a - |u|_b$$

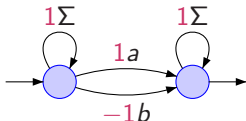
# Examples of weighted automata

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



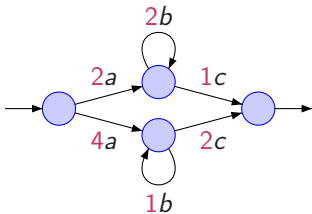
$$\llbracket \mathcal{A} \rrbracket(u) = 2^{|u|} \quad (\text{deterministic})$$

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



$$\llbracket \mathcal{A} \rrbracket(u) = |u|_a - |u|_b$$

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



$$\llbracket \mathcal{A} \rrbracket(ab^n c) = \min(3 + 2n, 6 + n)$$

# Weighted automata cannot compute large weights

## Remark

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

$$\llbracket \mathcal{A} \rrbracket(u) = O(M^{|u|}).$$

- ▶ 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  $|u|$ :

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

# Weighted Expressions

## Syntax of $WE(\mathcal{L})$

Fix  $\mathcal{L}$  a logic (eg, MSO,  $FO(<)$ ).

$$E ::= \varphi \mid k \mid E \oplus E \mid E \otimes E \mid \bigoplus_x E \mid \bigotimes_x E$$

where  $\varphi \in \mathcal{L}$ ,  $k \in K$ ,  $x$  is a first-order variable.

## Semantics

- ▶ An expression  $E$  without free variables defines a mapping  $\llbracket \varphi \rrbracket : \Sigma^+ \rightarrow K$ .
- ▶ For  $\varphi \in \mathcal{L}$ , we have  $\llbracket \varphi \rrbracket(w) \in \{0, 1\}$  (in the chosen semiring).

First order variables are interpreted as positions in the word.

For  $a \in \Sigma$ ,  $P_a(x)$  means “position  $x$  carries an  $a$ ”.

$x \leq y$  means “position  $x$  is before position  $y$ ”.

# Weighted Expressions

## Syntax of $WE(\mathcal{L})$

Fix  $\mathcal{L}$  a logic (eg, MSO,  $FO(<)$ ).

$$E ::= \varphi \mid k \mid E \oplus E \mid E \otimes E \mid \bigoplus_x E \mid \bigotimes_x E$$

where  $\varphi \in \mathcal{L}$ ,  $k \in K$ ,  $x$  is a first-order variable.

## Semantics

- ▶ An expression  $E$  without free variables defines a mapping  $\llbracket \varphi \rrbracket : \Sigma^+ \rightarrow K$ .
- ▶ For  $\varphi \in \mathcal{L}$ , we have  $\llbracket \varphi \rrbracket(w) \in \{0, 1\}$  (in the chosen semiring).

First order variables are interpreted as positions in the word.

For  $a \in \Sigma$ ,  $P_a(x)$  means “position  $x$  carries an  $a$ ”.

$x \leq y$  means “position  $x$  is before position  $y$ ”.

- ▶  $\bigoplus_x \varphi$  interpreted as a **sum** over all positions.
- ▶  $\bigotimes_x \varphi$  interpreted as a **product** over all positions.

## Weighted expressions: examples

▶  $\llbracket \bigoplus_x P_a(x) \rrbracket(u) = \sum_{i \in \text{pos}(u)} \llbracket P_a(x) \rrbracket(u, i) = |u|_a$

recognizable

## Weighted expressions: examples

- ▶  $\llbracket \bigoplus_x P_a(x) \rrbracket(u) = \sum_{i \in \text{pos}(u)} \llbracket P_a(x) \rrbracket(u, i) = |u|_a$  recognizable
- ▶  $\llbracket \bigotimes_y 2 \rrbracket(u) = \prod_{i \in \text{pos}(u)} \llbracket 2 \rrbracket(u, i) = 2^{|u|}$  recognizable



## Weighted expressions: examples

- ▶  $\llbracket \bigoplus_x P_a(x) \rrbracket(u) = \sum_{i \in \text{pos}(u)} \llbracket P_a(x) \rrbracket(u, i) = |u|_a$  recognizable
- ▶  $\llbracket \bigotimes_y 2 \rrbracket(u) = \prod_{i \in \text{pos}(u)} \llbracket 2 \rrbracket(u, i) = 2^{|u|}$  recognizable
- ▶  $\llbracket \bigotimes_x \bigotimes_y 2 \rrbracket(u) = \prod_{i \in \text{pos}(u)} \llbracket \bigotimes_y 2 \rrbracket(u, i) = (2^{|u|})^{|u|} = 2^{|u|^2}$ . not recognizable

w-Automata are not closed under universal quantification.

## Weighted expressions: examples

▶  $\llbracket \bigoplus_x P_a(x) \rrbracket(u) = \sum_{i \in \text{pos}(u)} \llbracket P_a(x) \rrbracket(u, i) = |u|_a$  recognizable

▶  $\llbracket \bigotimes_y 2 \rrbracket(u) = \prod_{i \in \text{pos}(u)} \llbracket 2 \rrbracket(u, i) = 2^{|u|}$  recognizable

▶  $\llbracket \bigotimes_x \bigotimes_y 2 \rrbracket(u) = \prod_{i \in \text{pos}(u)} \llbracket \bigotimes_y 2 \rrbracket(u, i) = (2^{|u|})^{|u|} = 2^{|u|^2}$ . not recognizable

w-Automata are not closed under universal quantification.

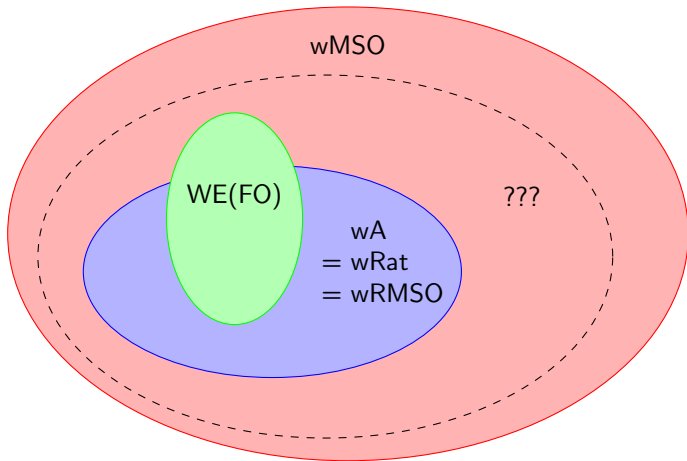
### Theorem (Droste & Gastin'05)

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

wRMSO consists of weighted expressions with

- ▶  $\bigotimes_x$  restricted to  $\vee \wedge$  of constants and boolean formulae.
- ▶ A new second order weighted operator,  $\bigoplus_x$ , restricted to boolean formulae.

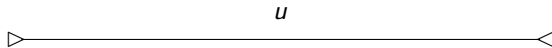
# Extending instead of Restricting ?



Aim: robust class extending both  $WE(FO)$  and  $wAutomata$ .

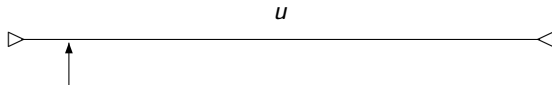
# (2-way) Pebble weighted automata

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



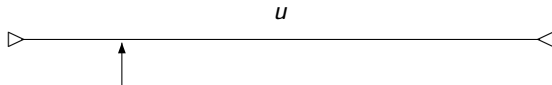
# (2-way) Pebble weighted automata

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



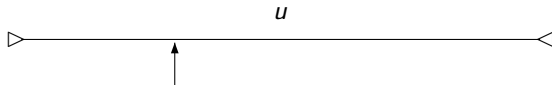
# (2-way) Pebble weighted automata

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



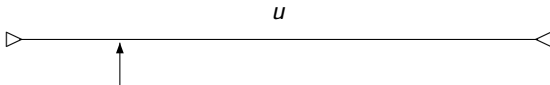
# (2-way) Pebble weighted automata

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



# (2-way) Pebble weighted automata

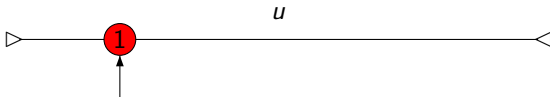
- ▶ Automaton with 2-way mechanism and pebbles  $\{1, \dots, r\}$ .





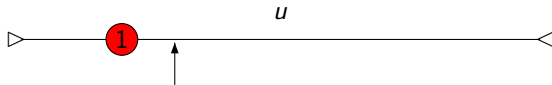
# (2-way) Pebble weighted automata

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



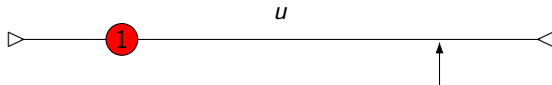
# (2-way) Pebble weighted automata

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



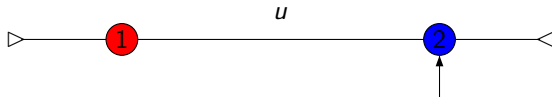
# (2-way) Pebble weighted automata

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



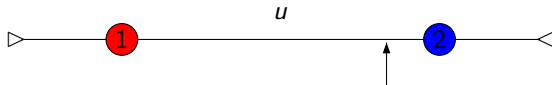
# (2-way) Pebble weighted automata

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



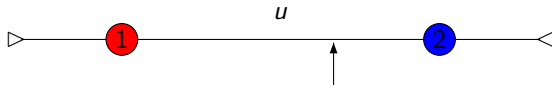
# (2-way) Pebble weighted automata

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



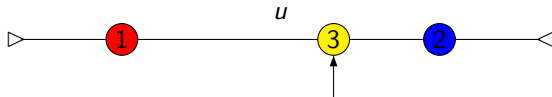
# (2-way) Pebble weighted automata

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



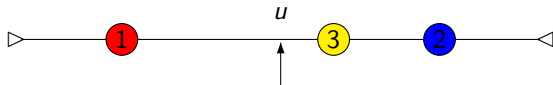
# (2-way) Pebble weighted automata

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



## (2-way) Pebble weighted automata

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

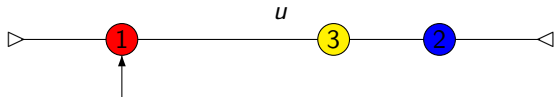


- ▶ Applicable transitions depend on current (state, letter, pebbles).  
( $p, ka, \text{Pebbles}, D, q$ ), where  $D \in \{\leftarrow, \rightarrow, \text{lift}, \text{drop}\}$ .



## (2-way) Pebble weighted automata

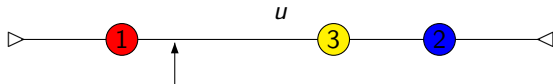
- ▶ Automaton with 2-way mechanism and pebbles  $\{1, \dots, r\}$ .



- ▶ 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

## (2-way) Pebble weighted automata

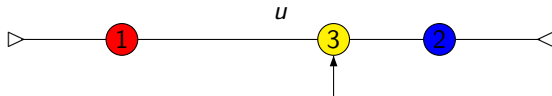
- ▶ Automaton with 2-way mechanism and pebbles  $\{1, \dots, r\}$ .



- ▶ 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.

## (2-way) Pebble weighted automata

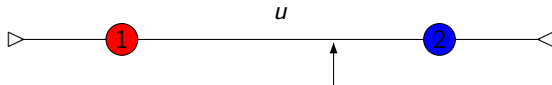
- ▶ Automaton with 2-way mechanism and pebbles  $\{1, \dots, r\}$ .



- ▶ 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.

## (2-way) Pebble weighted automata

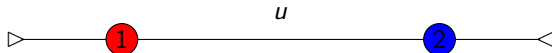
- ▶ Automaton with 2-way mechanism and pebbles  $\{1, \dots, r\}$ .



- ▶ 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.

# (2-way) Pebble weighted automata

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



- ▶ 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.
- ▶ **Note**. For Boolean word automata, this does not add expressive power.

# Pebble weighted automata: semantics

Recall from the classical setting:

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

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

- ▶ No longer well defined for 2-way pebble automata  
(can loop  $\Rightarrow$  can have arbitrarily large runs on a given word).

# Pebble weighted automata: semantics

Recall from the classical setting:

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

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

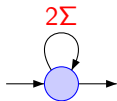
- ▶ No longer well defined for 2-way pebble automata (can loop  $\Rightarrow$  can have arbitrarily large runs on a given word).
- ▶ Value of a word: sum of all weights of **simple** runs on this word.

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

- ▶ (Other solution would be to restrict to suitable semirings)

# Why pebbles?

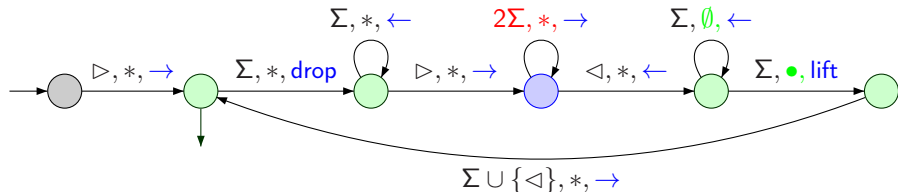
- ▶ Intuitively, pebbles can be used to encode a variable (at its position).





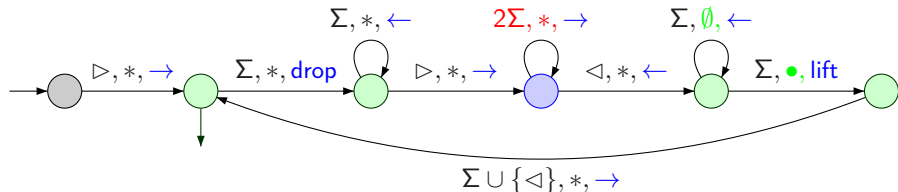
# Why pebbles?

- ▶ Intuitively, pebbles can be used to encode a variable (at its position).



# Why pebbles?

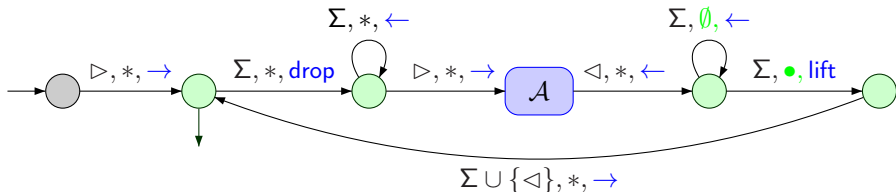
- ▶ Intuitively, pebbles can be used to encode a variable (at its position).



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

# Why pebbles?

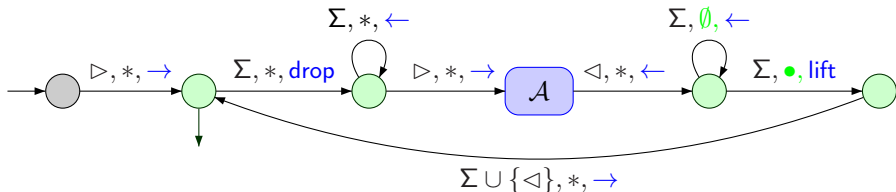
- ▶ Intuitively, pebbles can be used to encode a variable (at its position).



- ▶ Computes  $2^{|u|^2}$ : pebbles **add expressive power**.
- ▶ Very same idea: pebble weighted automata are closed under  $\otimes_x$ .

# Why pebbles?

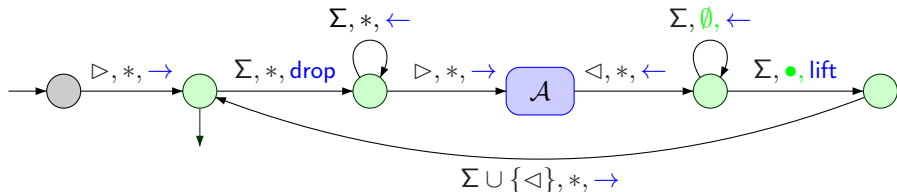
- ▶ Intuitively, pebbles can be used to encode a variable (at its position).



- ▶ Computes  $2^{|u|^2}$ : pebbles **add expressive power**.
- ▶ Very same idea: pebble weighted automata are closed under  $\otimes_x$ .
- ▶ Closure under  $\oplus_x$  by dropping **nondeterministically** the pebble instead.

## Why pebbles?

- ▶ Intuitively, pebbles can be used to encode a variable (at its position).



- ▶ Computes  $2^{|u|^2}$ : pebbles **add expressive power**.
- ▶ Very same idea: pebble weighted automata are closed under  $\otimes_x$ .
- ▶ Closure under  $\oplus_x$  by dropping **nondeterministically** the pebble instead.
- ▶ Summary: pebble wA easily bring **closure under  $\otimes_x$  and  $\oplus_x$** .

# 1-way pebble weighted automata

The construction for closure under  $\oplus_x$  and  $\otimes_x$  uses specific automata.

- ▶ After a **drop**, go to the end of the word and **reset**.
- ▶ No other use of  $\leftarrow$  **move**.

## 1-way pebble automata with $\ell$ -resets

A 1-way pebble automaton is a 2-way pebble automaton st.

- ▶ no  $\leftarrow$  move. Replaced by new move: **reset**.
- ▶ no **lift** can be immediately followed by a **drop**,
- ▶ each time a pebble is dropped, it gets a credit for  $\ell$  resets (recursively).

Similar to 1-way automata used by Neven, Schwentick, Vianu 04 for data words.

# Closure properties of 1 way pwA

- ▶ 1-way pebble automata are nonlooping (pebble are “progressing”).
- ▶ Semantics well-defined as  $\llbracket \mathcal{A} \rrbracket(w) = \sum_{\rho \text{ run of } \mathcal{A} \text{ on } w} \text{weight}(\rho)$ .
- ▶ Conditions can be enforced from a 2-way automaton by synchronizing it with a universal 1-way pebble automaton with  $\ell$ -resets.
- ▶ Still closed under  $\bigoplus_x$  and  $\bigotimes_x$ .
- ▶ Closed under  $\oplus$  and  $\otimes$ .
  - ▶ easy thanks to resets.
  - ▶ actually, resets not needed in commutative semirings.

# Closure properties of 1 way pwA

- ▶ 1-way pebble automata are nonlooping (pebble are “progressing”).
- ▶ Semantics well-defined as  $\llbracket \mathcal{A} \rrbracket(w) = \sum_{\rho \text{ run of } \mathcal{A} \text{ on } w} \text{weight}(\rho)$ .
- ▶ Conditions can be enforced from a 2-way automaton by synchronizing it with a universal 1-way pebble automaton with  $\ell$ -resets.
- ▶ Still closed under  $\bigoplus_x$  and  $\bigotimes_x$ .
- ▶ Closed under  $\oplus$  and  $\otimes$ .
  - ▶ easy thanks to resets.
  - ▶ actually, resets not needed in commutative semirings.

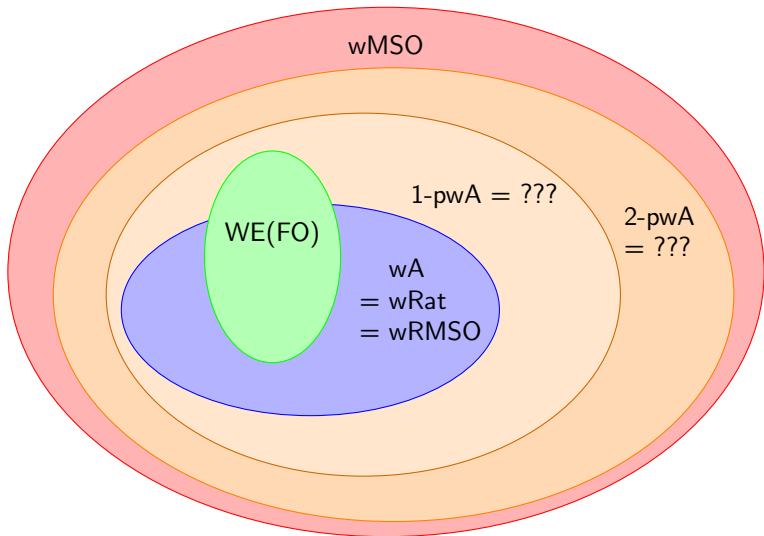
## Corollary

1-way pebble weighted automata capture WE(MSO).

For the converse, need to enrich the weighted expressions



# Pebble weighted Automata vs WE(MSO)



Characterization of 1-way and 2-way pebble  $wA$  in terms of expressions?

# Weighted transitive closure: TC and BTC

- ▶ Boolean setting:  $\text{MSO} \equiv \text{FO}(<) + \text{Transitive closure}$ .
- ▶ Intuition for a weighted transitive closure operator: **path costs**.
- ▶ Let  $w \in \Sigma^*$  and  $E(x, y) \in \text{WE}(\text{MSO})$  encoding a weighted graph  $G$ :
  - ▶ Set of vertices  $\text{Pos}(w)$ ,
  - ▶ Set of edges  $\{(x, y) \in V^2 \mid \llbracket E \rrbracket(x, y) \neq 0\}$
  - ▶ Costs defined by  $\lambda(x, y) = \llbracket E \rrbracket(x, y)$ .

Weighted transitive closure should compute, given vertices  $i, j$ , the **sum of the weights of paths** leading from  $i$  to  $j$  in  $G$ .

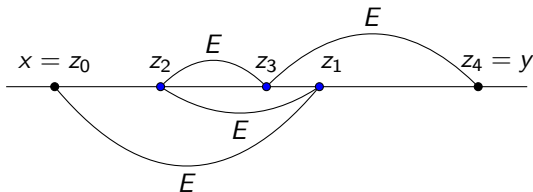
- ▶ Several natural interpretations according to the semiring.

## Weighted transitive closure: TC and BTC

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

$$E^n(x, y) = \sum_{x=z_0, z_1, \dots, z_n=y} \left( \prod_{1 \leq \ell \leq n} E(z_{\ell-1}, z_\ell) \right)$$

where the sum ranges over sequence of pairwise distinct positions  $z_0, \dots, z_n$ .

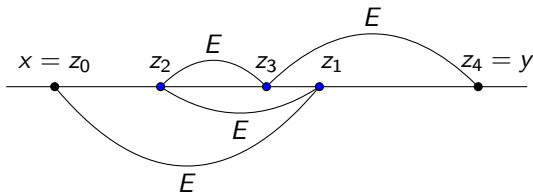


## Weighted transitive closure: TC and BTC

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

$$E^n(x, y) = \sum_{x=z_0, z_1, \dots, z_n=y} \left( \prod_{1 \leq \ell \leq n} E(z_{\ell-1}, z_\ell) \right)$$

where the sum ranges over sequence of pairwise distinct positions  $z_0, \dots, z_n$ .



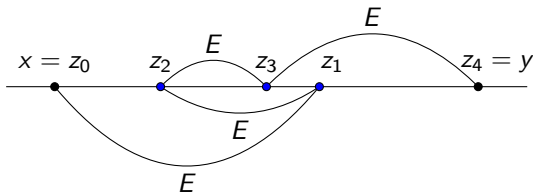
- ▶ The transitive closure operator is defined by  $TC_{xy} E = \bigvee_{n \geq 1} E^n$ .

## Weighted transitive closure: TC and BTC

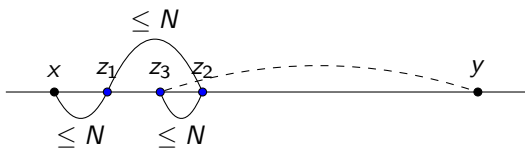
- ▶ For  $E(x, y)$  with (at least) two first order free variables, define

$$E^n(x, y) = \sum_{x=z_0, z_1, \dots, z_n=y} \left( \prod_{1 \leq \ell \leq n} E(z_{\ell-1}, z_\ell) \right)$$

where the sum ranges over sequence of pairwise distinct positions  $z_0, \dots, z_n$ .



- ▶ The transitive closure operator is defined by  $TC_{xy} E = \bigvee_{n \geq 1} E^n$ .
- ▶ Bounded transitive closure :  $N\text{-}TC_{xy} E = TC_{xy} (E \wedge |x - y| \leq N)$



# Bounded transitive closure and pebble automata

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

Given  $p$ -pebble automaton  $\mathcal{A}$  on  $\Sigma_{xy} = \Sigma \times \{0, 1\} \times \{0, 1\}$  recognizing  $\llbracket E \rrbracket(x, y)$  and a word  $(u, i, j)$



# Bounded transitive closure and pebble automata

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

Given  $p$ -pebble automaton  $\mathcal{A}$  on  $\Sigma_{xy} = \Sigma \times \{0, 1\} \times \{0, 1\}$  recognizing  $\llbracket E \rrbracket(x, y)$  and a word  $(u, i, j)$



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

# Bounded transitive closure and pebble automata

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

Given  $p$ -pebble automaton  $\mathcal{A}$  on  $\Sigma_{xy} = \Sigma \times \{0, 1\} \times \{0, 1\}$  recognizing  $\llbracket E \rrbracket(x, y)$  and a word  $(u, i, j)$



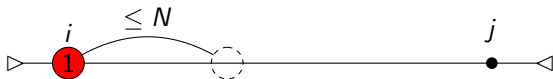
1.  $\mathcal{B}$  goes to  $i$  and drops **pebble 1**
2.  $\mathcal{B}$  drops nondeterministically **pebble 2** on a position at distance  $\leq N$
3.  $\mathcal{B}$  simulates  $\mathcal{A}$  on  $w$  where  $x$  and  $y$  are mapped to the positions of pebbles



# Bounded transitive closure and pebble automata

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

Given  $p$ -pebble automaton  $\mathcal{A}$  on  $\Sigma_{xy} = \Sigma \times \{0, 1\} \times \{0, 1\}$  recognizing  $\llbracket E \rrbracket(x, y)$  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  $N\text{-TC}_{xy}E$  with 2 additional pebbles:

Given  $p$ -pebble automaton  $\mathcal{A}$  on  $\Sigma_{xy} = \Sigma \times \{0, 1\} \times \{0, 1\}$  recognizing  $\llbracket E \rrbracket(x, y)$  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

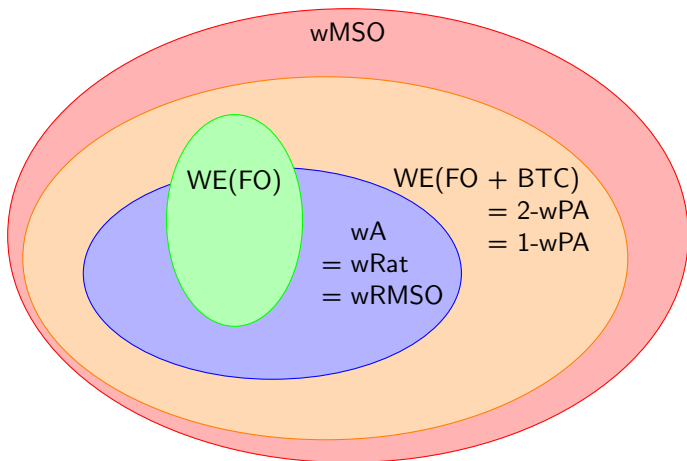
## Theorem (BGMZ'10)

$$\text{WE}(\text{FO} + \text{BTC}^<) = \text{2-way pebble wA} = \text{1-way pebble wA}$$

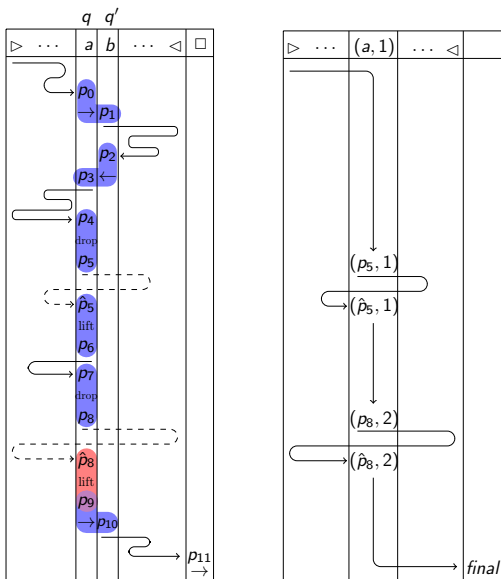
- ▶ Proof of  $\text{WE}(\text{FO} + \text{BTC}^<) \subseteq \text{2-way pebble wA}$  done in the previous slides.
- ▶ Proof of  $\text{2-way pebble wA} \subseteq \text{1-way pebble wA}$ :  
Generalization of the translation of 2-way automata to 1-way automata.
- ▶ Proof of  $\text{1-way pebble wA} \subseteq \text{WE}(\text{FO} + \text{BTC}^<)$ :  
Generalization of a proof showing that weighted automata are expressible with transitive closure.

# Summary

- ▶ Pebbles and 1-way pebbles add expressive power in weighted automata.
- ▶ Negative result: SAT of  $WE(FO + BTC^<)$  is not decidable in general.



# Flavor of the proof of $1\text{-pebble} \subseteq 1\text{-nested}$



Requires commutativity

# SAT for 2-way pebble wA

## Undecidability of SAT

SAT is undecidable for 2-way pebble wA, with 2 pebbles over  $(\mathbb{Z}, +, \times)$ .

- ▶ Given  $\mathcal{A}$ , is there a word such that  $\llbracket \mathcal{A} \rrbracket \neq 0$ ?
- ▶ Reduction from halting problem for Minsky machines.
- ▶ From Minsky machine  $\mathcal{M}$ , build  $\mathcal{A}$  over  $(\mathbb{Z}, +, \times)$ 
  - ▶ which reads sequences of transitions of  $\mathcal{M}$ , in  $\{D_1, D_2, l_1, l_2, Z_1, Z_2\}^+$ .
  - ▶ assigns weight 0 to illegal runs, nonzero to legal ones.

# SAT for 2-way pebble wA

## Undecidability of SAT

SAT is undecidable for 2-way pebble wA, with 2 pebbles over  $(\mathbb{Z}, +, \times)$ .

- ▶ Given  $\mathcal{A}$ , is there a word such that  $\llbracket \mathcal{A} \rrbracket \neq 0$ ?
- ▶ Reduction from halting problem for Minsky machines.
- ▶ From Minsky machine  $\mathcal{M}$ , build  $\mathcal{A}$  over  $(\mathbb{Z}, +, \times)$ 
  - ▶ which reads sequences of transitions of  $\mathcal{M}$ , in  $\{D_1, D_2, I_1, I_2, Z_1, Z_2\}^+$ .
  - ▶ assigns weight 0 to illegal runs, nonzero to legal ones.
- ▶ 4 conditions to check that a run is illegal. Eg, for counter  $c^1$ :
  - ▶ counter  $c^1$  is zero at a **D**ecrement: compute  $\prod_{a_j=D_1} c_{j-1}^1(w)$   
Note:  $c_{j-1}^1(w)$  is just the difference between the numbers of  $I_1$ 's and  $D_1$ 's.
  - ▶ counter  $c^1$  is nonzero at a **Z**ero test: compute  $\prod_{a_j=Z_1} \prod_{k=1}^j [c_j^1(w) - k]$ .

# Related and further work

1. Algorithms for pebble wA (eg. model-checking)?

Emptiness: is  $\llbracket A \rrbracket = 0$  decidable?

Yes on positive semirings.

NO on fields (recall: emptiness decidable for wA from Schützenberger'62).

2. Unbounded steps in transitive closure?
3. Weak pebbles vs. strong pebbles?
4. Extended wRat for wPA?



## Open problems (2)

Extensions to other structures: Trees (ranked or unranked)

1. Tree walking automata (TWA) are 2-way automata
2. 1-way TWA = Depth First Search Automata (DFSA)
3. Some results go through:  $w\text{-DFSA} = w(\text{FO} + \text{BTC}^<)$  Additional technical difficulties:
  - ▶ On words: usual encoding to come back to expressions  $WE(\text{FO} + \text{BTC}^<)$  based on suitable factorization.
  - ▶ The  $w\text{DFS}$  automata follow a DFS walk but may nondeterministically cut subtrees.

## Open problems (2)

Extensions to other structures: Trees (ranked or unranked)

1. Tree walking automata (TWA) are 2-way automata
2. 1-way TWA = Depth First Search Automata (DFSA)
3. Some results go through:  $w\text{-DFSA} = w(\text{FO} + \text{BTC}^<)$  Additional technical difficulties:
  - ▶ On words: usual encoding to come back to expressions  $WE(\text{FO} + \text{BTC}^<)$  based on suitable factorization.
  - ▶ The  $w\text{DFS}$  automata follow a DFS walk but may nondeterministically cut subtrees.
4. pebble TWA  $\stackrel{?}{=} \text{pebble DFSA}$

## Open problems (2)

Extensions to other structures: Trees (ranked or unranked)

1. Tree walking automata (TWA) are 2-way automata
2. 1-way TWA = Depth First Search Automata (DFSA)
3. Some results go through:  $w\text{-DFSA} = w(\text{FO} + \text{BTC}^<)$  Additional technical difficulties:
  - ▶ On words: usual encoding to come back to expressions  $WE(\text{FO} + \text{BTC}^<)$  based on suitable factorization.
  - ▶ The  $w\text{DFS}$  automata follow a DFS walk but may nondeterministically cut subtrees.
4. pebble TWA  $\stackrel{?}{=} \text{pebble DFSA}$
5. Quantitative query languages:  $w\text{XPath}$ ,  $w\text{RXPath}$