Specification and Verification of Quantitative Properties: Expressions, Logics, and Automata

Paul GASTIN - LSV - ENS Cachan

05/05/2014

Joint work with Benedikt Bollig, Benjamin Monmege and Marc Zeitoun
Model Checking and Evaluation

Model: $G$
Computation Document System

Specification: $\Phi$
First-Order Logic
Temporal Logic
Propositional Dynamic Logic
Regular Expressions

$G \models \Phi$ Boolean
Model Checking and Evaluation

- May an error state be reached?
- Is there a book written by X, rented by Y?
- Does this leader election protocol permit to elect the leader?

From Boolean to Quantitative Verification

- What is the probability for an error state to be reached?
- How many books, written by X, have been rented by Y?
- What is the maximal delay ensuring that this leader election protocol permits the election?

\[ G \models \Phi \]

\[ \llbracket \Phi \rrbracket (G) \]

Boolean

Quantitative
Evaluation via Automata

Model: $G$

Specification: $\Phi$

Complexity?
Small specifications
Huge models

Automaton: $A_\Phi$

$G \in L(A_\Phi)$
$[A_\Phi](G)$
$G \models \Phi$
$[\Phi](G)$

Boolean
Quantitative
The Success Story of Automata

An Automata-Theoretic Approach to ...

... Parsing Regular Expressions
Kleene’s theorem [Kleene 56]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]


... Parsing Weighted Expressions
Schützenberger’s theorem
[Schützenberger 61]

... Deciding MSO Over Finite Words
[Büchi 59, Elgot 61, Trakhtenbrot 61]

... Model Checking a System Against LTL
[Vardi&Wolper 86]
MODELS
Various Models

• Words

Computations of sequential programs

\[ a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow b \rightarrow b \]

• Nested Words

Computations of recursive programs

XML documents

```plaintext
proc f ()
{ ... }

proc g ()
{ ... }

main (n)
{
    i=0;
    while i<n do
        if i odd then f() else g()
        i++
    done
}
```
Various Models

- **Words**
  - Computations of sequential programs
  - $a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow b \rightarrow b$

- **Nested Words**
  - Computations of recursive programs
  - XML documents
  - $a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow b \rightarrow b$

- **Ranked Trees**
  - Expressions, Formulas, Parse trees, ...

- **Pictures**
Various Models

• **Mazurkiewicz Traces**
  
  Computations of concurrent programs
  Communication by Rendez-Vous

• **Traces with Nestings**
  
  Concurrent and Recursive Programs
  Communication by Rendez-Vous

• **Message Sequence Charts**
  
  Communication by FIFO channels
Various Models

• Mazurkiewicz Traces

  Computations of concurrent programs
  Communication by Rendez-Vous

  ![Mazurkiewicz traces diagram]

• Traces with Nestings

  Concurrent and Recursive Programs
  Communication by Rendez-Vous

  ![Traces with nestings diagram]

• Message Sequence Charts with Nestings

  Communication by FIFO channels

  ![Message sequence charts with nestings diagram]
Modeling a picture as a graph
Modeling a picture as a graph
Graphs

\[ G = (V, (E_d)_{d \in D}, \lambda) \]

- \( V \): set of vertices
- \( \lambda \): labels of vertices
- \( D \): set of directions
- \( E_d \): set of \( d \)-edges

Deterministic (hence bounded degree)

\[ D = \{ \rightarrow, \downarrow \} \cup \{ \leftarrow, \uparrow \} \]
Various Models

• **Words**
  Computations of sequential programs
  
  \[ D = \{ \rightarrow, \leftarrow \} \]

• **Nested Words**
  Computations of recursive programs
  XML documents
  
  \[ D = \{ \rightarrow, \leftarrow, \rhd, \lhd \} \]

• **Ranked Trees**
  Expressions, Formulas, Parse trees, ...
  
  \[ D = \{ \downarrow_1, \uparrow_1, \downarrow_2, \uparrow_2 \} \]

• **Pictures**
  
  \[ D = \{ \rightarrow, \leftarrow, \downarrow, \uparrow \} \]
Weighted First-Order Logic
Logical Specifications: Query Examples

Is there a line of green pixels?

\[ \exists x \forall y \left[ (R_{\rightarrow}(x, y) \lor R_{\rightarrow}^{*}(y, x)) \Rightarrow P_{\square}(y) \right] \]

How many lines of green pixels are there?

What is the size of the picture?

Boolean fragment: first-order logic

\[ \varphi ::= \top \mid P_a(x) \mid x = y \mid R_d(x, y) \mid R_d^{+}(x, y) \mid \neg \varphi \mid \varphi \lor \varphi \mid \exists x \varphi \]

with \( a \in A \cup \{ \triangleright \} \), \( d \in D \).

What is the size of the biggest monochromatic rectangle?
Logical Specifications: Query Examples

Is there a line of green pixels?

$$\exists x \forall y \left[ (R_{\rightarrow}^*(x, y) \lor R_{\rightarrow}^*(y, x)) \Rightarrow P_\square(y) \right]$$

How many lines of green pixels are there?

$$\sum_x \left( \neg \exists y \ R_{\rightarrow}(y, x) \right) \land \left( \forall y \ R_{\rightarrow}^*(x, y) \Rightarrow P_\square(y) \right) \ ? \ 1 : 0$$

What is the size of the picture?

Weighted fragment: first-order logic

$$\Phi ::= s \mid \varphi \ ? \ \Phi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus_x \Phi \mid \bigotimes_x \Phi$$

with $$\varphi \in bFO$$ and $$s \in S$$.

What is the size of the biggest monochromatic rectangle?
Logical Specifications: Query Examples

Is there a line of green pixels?
\[ \exists x \forall y \left[ (R_x^\rightarrow (y, x) \lor R_y^\rightarrow (y, x)) \Rightarrow P(y) \right] \]

How many lines of green pixels are there?
\[ \sum_x (\neg \exists y R_{\rightarrow} (y, x)) \land (\forall y R_{\rightarrow}^*(x, y) \Rightarrow P(y)) ? 1 : 0 \]

What is the size of the picture?
\[ \left( \sum_x \neg \exists y R_{\rightarrow} (y, x) \right) \times \left( \sum_x \exists y R_{\rightarrow} (x, y) \right) \]

What is the color with maximum number of pixels?
\[ \max \left( \sum_x P_{\square} (x), \sum_x P_{\black} (x), \ldots, \sum_x P_{\triangle} (x) \right) \]

What is the size of the biggest monochromatic rectangle?
\[ \max_{x, y} \left[ \varphi_{\text{mono}}(x, y) ? \left( \sum_z \varphi_{\text{rect}}(x, y, z) ? 1 : 0 \right) : -\infty \right] \]
Full Weighted First-order Logic

**Boolean fragment**

\[ \phi ::= \top \mid P_a(x) \mid x = y \mid R_d(x, y) \mid R^+_d(x, y) \mid \neg \phi \mid \phi \lor \phi \mid \exists x \phi \]

**Quantitative formulas**

\[ \Phi ::= s \mid \phi?\Phi : \Phi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus_x \Phi \mid \bigotimes_x \Phi \]

with \( a \in A \cup \{\triangleright\} \), \( d \in D \), \( s \in S \).

**Path using d-edges**

For trees we may use \( \downarrow^+ \)

**Possible Variants:**
- change Boolean power (AP, MSO, ...)
- add more quantitative power (wMSO, wTC...)
Quantitative Semantics

$$[[s]](G, \sigma) = s$$

$$[[\varphi ? \Phi_1 : \Phi_2]](G, \sigma) = \begin{cases} 
[[\Phi_1]](G, \sigma) & \text{if } G, \sigma \models \varphi \\
[[\Phi_2]](G, \sigma) & \text{otherwise}
\end{cases}$$

$$[[\Phi_1 \oplus \Phi_2]](G, \sigma) = [[\Phi_1]](G, \sigma) + [[\Phi_2]](G, \sigma)$$

$$[[\Phi_1 \otimes \Phi_2]](G, \sigma) = [[\Phi_1]](G, \sigma) \times [[\Phi_2]](G, \sigma)$$

$$[[\bigoplus_x \Phi]](G, \sigma) = \sum_{k \in \text{pos}(G)} [[\Phi]](G, \sigma[x \mapsto k])$$

$$[[\bigotimes_x \Phi]](G, \sigma) = \prod_{k \in \text{pos}(G)} [[\Phi]](G, \sigma[x \mapsto k])$$

Weights from a semiring $$(S, +, \times, 0, 1)$$

Valuation of free variables $\sigma : V \rightarrow \text{pos}(G)$. 

Paul Gastin, LSV, ENS de Cachan

WATA, Leipzig, May 5th 2014
Quantitative Semantics

Weights from a semiring $(\mathcal{S}, +, \times, 0, 1)$

\[
[s](G, \sigma) = s
\]

\[
[\varphi ? \Phi_1 : \Phi_2](G, \sigma) = \begin{cases} 
[\Phi_1](G, \sigma) & \text{if } G, \sigma \models \varphi \\
[\Phi_2](G, \sigma) & \text{otherwise}
\end{cases}
\]

\[
[\Phi_1 \oplus \Phi_2](G, \sigma) = [\Phi_1](G, \sigma) + [\Phi_2](G, \sigma)
\]

\[
[\Phi_1 \otimes \Phi_2](G, \sigma) = [\Phi_1](G, \sigma) \times [\Phi_2](G, \sigma)
\]

\[
[\bigoplus_x \Phi](G, \sigma) = \sum_{k \in \text{pos}(G)} [\Phi](G, \sigma[x \mapsto k])
\]

\[
[\bigotimes_x \Phi](G, \sigma) = \prod_{k \in \text{pos}(G)} [\Phi](G, \sigma[x \mapsto k])
\]
Weight Domains: Semirings

\((\mathbb{S}, +, \times, 0, 1)\)

- associative and commutative, with neutral element 0

- associative, with neutral element 1, distributive over addition

\((\mathbb{R}, +, \times, 0, 1)\)
\((\mathbb{Q}, +, \times, 0, 1)\)
\((\mathbb{Z}, +, \times, 0, 1)\)
\((\mathbb{N}, +, \times, 0, 1)\)

\([\{0, 1\}, \lor, \land, 0, 1]\)
\(([0, 1], \max, \min, 0, 1)\)
\((\mathbb{R} \cup \{-\infty, +\infty\}, \max, \min, -\infty, +\infty)\)

\(\mathbb{B}\langle A^* \rangle = (\mathcal{P}(A^*), \cup, \cdot, \emptyset, \{\varepsilon\})\)
\(\mathbb{N}\langle A^* \rangle = (\mathbb{N} A^*, +, \cdot, 0, 1_\varepsilon)\)
\((\mathbb{R} \cup \{+\infty\}, \min, +, +\infty, 0)\)
\((\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)\)
Restricted Weighted MSO Logic

Booolean fragment

\[ \varphi ::= \top | P_a(x) | x = y | R_d(x,y) | R_d^+(x,y) | \neg \varphi | \varphi \lor \varphi | \exists x \varphi | x \in X | \exists X \varphi \]

\[ \Phi ::= s | \varphi? \Phi : \Phi | \Phi \oplus \Phi | \bigoplus_x \Phi | \bigodot_x \Psi | \bigoplus_X \Phi \]

with \( a \in A \cup \{\triangleleft\} \), \( d \in D \), \( s \in S \).

Quantitative formulas

Almost boolean

\[ \Psi ::= s | \varphi? \Psi : \Psi \]

Droste&Gastin 07

Restricted wMSO = weighted automata
Restricted Weighted MSO Logic

### Boolean fragment

$$\varphi ::= \top \mid P_a(x) \mid x = y \mid R_d(x, y) \mid P_d(x, y) \mid \neg \varphi \mid \varphi \vee \varphi \mid \exists x \varphi \mid x \in X \mid \exists X \varphi$$

### commutative semiring

$$\Phi ::= s \mid \varphi? \Phi : \Phi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus_x \Phi \mid \bigotimes_x \Psi \mid \bigoplus_X \Phi$$

with $a \in A \cup \{\triangleright\}$, $d \in D$, $s \in S$.

### Quantitative formulas

### Almost boolean

$$\Psi ::= s \mid \varphi? \Psi : \Psi$$

Droste & Gastin 07

Restricted wMSO = weighted automata
Semirings vs Average discounted sum
Uninterpreted Weights

Do not interpret weights from $W$

Semiring of multi-sets of weight sequences

$S = \mathbb{N} \langle W^* \rangle = (\mathbb{N}^{W^*}, +, \cdot, 0, 1_\varepsilon)$

Restricted wMSO = weighted automata
Full wFO = pebble weighted automata
Uninterpreted Weights

Do not interpret weights from $W$

Semiring of multi-sets of weight sequences

$$S = \mathbb{N}\langle\langle W^* \rangle\rangle = (\mathbb{N}^{W^*}, +, \cdot, 0, 1_\varepsilon)$$

Valuations yields multi-sets of values

- **Average**: $\text{val}(w_1w_2\cdots w_n) = \frac{w_1+w_2+\cdots+w_n}{n}$
- **Discounted sum**: $\text{val}_\lambda(w_1w_2\cdots w_n) = \lambda^1w_1 + \lambda^2w_2 + \cdots + \lambda^n w_n$
- **Cost-Reward**: $\text{val}((r_1,c_1)(r_2,c_2)\cdots (r_n, c_n)) = \frac{r_1+r_2+\cdots+r_n}{c_1+c_2+\cdots+c_n}$
- **Probabilities**: $\text{val}((p_1,w_1)(p_2,w_2)\cdots (p_n, w_n)) = (p_1p_2\cdots p_n, w_1 + w_2 + \cdots w_n)$

Aggregation of multi-sets of values

$$\mathbb{N}^{S} \rightarrow T$$

Max, Min, Sum, Average, Expectation, …
Walking Weighted Automata with Pebbles
Pebble Walking Weighted Automata

\[ (\mathbb{Z} \cup \{-\infty\}, \max, +, -\infty, 0) \]

\[ \gamma = \sum_c (+c)e? \]

Pebble automata over words and trees: [Globerman&Harel 96], [Engelfriet&Hoogeboom 99]
Pebble Walking Weighted Automata

\[ (\mathbb{Z} \cup \{-\infty\}, \max, +, -\infty, 0) \]

\[ \gamma = \sum_c (c+e)c? \]

Pebble automata over words and trees: [Globerman & Harel 96], [Engelfriet & Hoogeboom 99]
Pebble Walking Weighted Automata

\[ \gamma = \sum_c (+c) \in \{ \infty \}, \text{max}, +, -\infty, 0 \]

\[ + 255 \]

\[ \text{Pebble automata over words and trees: [Globerman&Harel 96], [Engelfriet&Hoogeboom 99]} \]
Pebble Walking Weighted Automata

\[ \gamma = \sum_c (\pm c)e \]

\[ (\mathbb{Z} \cup \{-\infty\}, \max, +, -\infty, 0) \]

\[ + 255 - 0 \]

Pebble automata over words and trees: [Globerman&Harel 96], [Engelfriet&Hoogeboom 99]
Pebble Walking Weighted Automata

\[ \gamma = \sum_c (\pm c) e? \]

\[ \begin{align*}
\text{drop}_x & \quad \rightarrow + \downarrow \\
\text{drop}_y & \quad \rightarrow + \downarrow \\
\text{lift} & \quad \rightarrow + \downarrow
\end{align*} \]

\[ \begin{align*}
+\gamma(\rightarrow + \downarrow) & \quad \rightarrow + \downarrow \\
-\gamma(\rightarrow + \downarrow) & \quad \rightarrow + \downarrow \\
+\gamma & \quad \rightarrow + \downarrow \\
-\gamma & \quad \rightarrow + \downarrow
\end{align*} \]

\[ \begin{align*}
+x\land y & \quad ?\uparrow \\
(x\land ?) & \quad ?\uparrow \\
? & \quad ?\rightarrow \\
\end{align*} \]

\[ \begin{align*}
20 & \quad (Z \cup \{-\infty\}, \max, +, -\infty, 0) \\
+ 255 - 0 & \quad \text{Pebble automata over words and trees: [Globerman\&Harel 96], [Engelfriet\&Hoogeboom 99]} \]

WATA, Leipzig, May 5th 2014
Pebble Walking Weighted Automata

\( (\mathbb{Z} \cup \{ -\infty \}, \max, +, -\infty, 0) \)

\[ + 255 - 0 \]

\[ \gamma = \sum_c (+c)c? \]

\[ \text{Pebble automata over words and trees: [Globerman\&Harel 96], [Engelfriet\&Hoogeboom 99]} \]
Pebble Walking Weighted Automata

\[ (\mathbb{Z} \cup \{-\infty\}, \max, +, -\infty, 0) \]

+ 255 - 0

Pebble automata over words and trees: [Globerman&Harel 96], [Engelfriet&Hoogeboom 99]
Pebble Walking Weighted Automata

\[(\mathbb{Z} \cup \{-\infty\}, \max, +, -\infty, 0)\] 

\[+ 255 - 0\] 

\(\gamma = \sum_{c} (+c)c?\)

Pebble automata over words and trees: [Globerman&Harel 96], [Engelfriet&Hoogeboom 99]
Pebble Walking Weighted Automata

\[ \gamma = \sum_c (c+e)c? \]

\[ \text{drop}_x \]

\[ \text{drop}_y \]

\[ \text{lift} \]

\[ (Z \cup \{-\infty\}, \text{max}, +, -\infty, 0) \]

\[ + 255 - 0 \]

Pebble automata over words and trees: [Globerman & Harel 96], [Engelfriet & Hoogeboom 99]
Pebble Walking Weighted Automata

\[ \gamma = \sum_c (+c)c? \]

\[ \begin{align*}
\text{drop}_x & \quad +\gamma(\rightarrow + \downarrow) \quad -\gamma \rightarrow \quad \rightarrow + \downarrow \\
\text{drop}_y & \quad +\gamma \rightarrow \\
\text{lift} & \quad \rightarrow + \downarrow
\end{align*} \]

\[ (\mathbb{Z} \cup \{-\infty\}, \max, +, -\infty, 0) \]

\[ +255 - 0 \]

Pebble automata over words and trees: [Globerman&Harel 96], [Engelfriet&Hoogeboom 99]
Pebble Walking Weighted Automata

\[ \mathbb{Z} \cup \{-\infty\}, \max, +, -\infty, 0 \]

\[ + 255 - 0 \]

Weight of the run: 255

\[ \gamma = \sum_e (+e)e? \]

Pebble automata over words and trees: [Globerman&Harel 96], [Engelfriet&Hoogeboom 99]

Paul Gastin, LSV, ENS de Cachan

WATA, Leipzig, May 5th 2014
Pebble Walking Weighted Automata

Non determinism Resolved by max

Max of the weights of the runs: biggest contrast in a green rectangle

\[ \gamma = \sum_c (\pm c) e \]

\[ (\mathbb{Z} \cup \{-\infty\}, \max, +, -\infty, 0) \]

\[ + 255 - 0 \]

Weight of the run: 255

Pebble automata over words and trees: [Globerman & Harel 96], [Engelfriet & Hoogeboom 99]
Pebble Weighted Automata

\[ A = (Q, A, D, I, \Delta, F) \]

- finite set of states
- alphabet
- set of directions
- vector of initial weights
- matrix of transition
- vector of final weights

Transitions: linear combination of \((\text{test}, \text{action})\)

Boolean combination of \(T, a?, x?, d?, \text{init}?, \text{final}\)

Move \(d\),
Drop a pebble,
Lift the last dropped pebble
Semantics of Pebble Weighted Automata

Configuration of $\mathcal{A}$: $(G, q, \pi, v)$

- State: $G$
- Stack of dropped pebbles: $\pi$
- Current vertex: $q$

Run over a graph $G$: finite sequence of configurations

Weight of a run: multiplication of the weights of the transitions and the initial and final weights

Semantics $[\mathcal{A}](G)$: sum of the weights of the runs

Navigation possibly leads to infinitely many runs!

Restrict to continuous semirings
Weight Domains: **Continuous Semirings**

\[(\mathbb{S}, +, \times, 0, 1)\]

Every infinite sum exists and is the limit of finite *approximate* sums, keeping good properties of usual semiring

\[
\begin{align*}
(\mathbb{R}, +, \times, 0, 1) & \quad \text{Invalid} \\
(\mathbb{Q}, +, \times, 0, 1) & \quad \text{Invalid} \\
(\mathbb{Z}, +, \times, 0, 1) & \\
(\mathbb{N}, +, \times, 0, 1) & \quad \text{Invalid} \\
(\mathbb{P}(A^*), \cup, \cdot, \emptyset, \{\varepsilon\}) & \\
(\mathbb{R}^+ \cup \{+\infty\}, +, \times, 0, 1) & \\
(\mathbb{N} \cup \{+\infty\}, +, \times, 0, 1) & \\
(\mathbb{R} \cup \{-\infty, +\infty\}, \min, +, +\infty, 0) & \quad \text{Invalid} \\
(\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0) & \quad \text{Invalid}
\end{align*}
\]

\[([0, 1], \max, \min, 0, 1)\]

\[(\mathbb{R} \cup \{-\infty, +\infty\}, \max, \min, -\infty, +\infty)\]
Layered Automata

Layer 2:

Layer 1:

Layer 0:

Theorem:
For each \( K \)-layered pebble automaton \( A \), we can construct an equivalent pebble automaton \( E(A) \). Moreover, the pebble-depth of \( E(A) \) is at most \( K \).
Weighted FO and pebWA
wFO vs pebWA

Weighted First-Order logic

• Denotational semantics
• Boolean fragment
• Quantitative formulas

Weighted Automata with pebbles

• Operational semantics
• Only quantitative computations
wFO vs pebWA

Weighted First-Order logic

• Denotational semantics
• Boolean fragment
• Quantitative formulas

Weighted Automata with pebbles

• Operational semantics
• Only quantitative computations

Thm: Consider a searchable class of graphs.

For each wFO formula, we can construct an equivalent layered pebWA,
# states linear in the size of the formula
# layers = quantifier depth of the formula
# pebble names = # variable names in the formula
A class $\mathcal{G}$ of graphs is searchable if there exists a walking automaton $A_\mathcal{G}$ with two states $q_i$ and $q_f$ such that for each graph $G = (V, (R_d)_{d \in D}, \lambda, \triangleright)$ in $\mathcal{G}$

- There exists a total order $\leq$ on $E$ with minimal element $\triangleright$, 
- for every $v \in V$, there is a unique run of $A_\mathcal{G}$ starting from $(G, q_i, v)$ and ending in state $q_f$,
- this run ends in configuration $(G, q_0, v')$ with $v \preceq v'$ if $v$ is not maximal in $(V, \leq)$ and $v' = \triangleright$ otherwise.

The walking automaton $A_\mathcal{G}$ is the guide for the class $\mathcal{G}$.

- **Words**
  Computations of sequential programs
  $D = \{\rightarrow, \leftarrow\}$

- **Nested Words**
  Computations of recursive programs
  XML documents
  $D = \{\rightarrow, \leftarrow, \multimap, \lhd\}$

\[ a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow b \rightarrow b \]
A class $\mathcal{G}$ of graphs is searchable if there exists a walking automaton $A_{\mathcal{G}}$ with two states $q_i$ and $q_f$ such that for each graph $G = (V, (R_d)_{d \in D}, \lambda, \triangleright)$ in $\mathcal{G}$:

- There exists a total order $\leq$ on $E$ with minimal element $\triangleright$,
- for every $v \in V$, there is a unique run of $A_{\mathcal{G}}$ starting from $(G, q_i, v)$ and ending in state $q_f$,
- this run ends in configuration $(G, q_0, v')$ with $v \preceq v'$ if $v$ is not maximal in $(V, \leq)$ and $v' = \triangleright$ otherwise.

The walking automaton $A_{\mathcal{G}}$ is the guide for the class $\mathcal{G}$.

• Ranked Trees

Expressions, Formulas, Parse trees, ...

$D = \{\downarrow_1, \uparrow_1, \downarrow_2, \uparrow_2\}$

DFS

\[ \neg(\downarrow_1?) \quad \downarrow_2 \quad \text{root?} \]
\[ \neg(\downarrow_2?) \quad \uparrow_1 \quad \text{root?} \]
wFO to pebWA

$$\varphi ::= \top | P_a(x) | x = y | R_d(x, y) | R_d^+(x, y) | \neg \varphi | \varphi \lor \varphi | \exists x \varphi$$

$$\Phi ::= s | \varphi? \Phi : \Phi | \Phi \oplus \Phi | \Phi \otimes \Phi | \bigoplus \Phi | \bigotimes \Phi$$

with \(a \in A \cup \{\triangledown\}, \ d \in D, \ s \in S.\)

$$\Phi \oplus \Psi$$  disjoint union of automata

$$\Phi \otimes \Psi$$  with \(a \in A \cup \{\triangledown\}, \ d \in D, \ s \in S.\)

Runs end on the marker
Searchable graphs
wFO to pebWA

\[ \varphi ::= \top \mid P_a(x) \mid x = y \mid R_d(x, y) \mid R_d^+(x, y) \mid \neg \varphi \mid \varphi \lor \varphi \mid \exists x \varphi \]

\[ \Phi ::= s \mid \varphi? \Phi : \Phi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus \Phi \mid \bigotimes \Phi \mid \forall x \Phi \]

with \( a \in A \cup \{\triangleright\} \), \( d \in D \), \( s \in S \).

\[ \Phi \oplus \Psi \quad \text{disjoint union of automata} \]

\[ \Phi \otimes \Psi \quad \rightarrow \quad \mathcal{A}^\Phi \quad \rightarrow \quad \mathcal{A}^\Psi \]

\[ \varphi? \Phi : \Psi \quad \rightarrow \quad \mathcal{B}^\varphi \quad \rightarrow \quad \mathcal{A}^\Phi \quad \rightarrow \quad \mathcal{A}^\Psi \]

Runs end on the marker

Searchable graphs
wFO to pebWA

\[ \varphi ::= \top \mid P_a(x) \mid x = y \mid x \xrightarrow{d} y \mid x \xrightarrow{d^+} y \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \exists x \varphi \mid \forall x \varphi \]

\[ \Phi ::= s \mid \varphi? \Phi : \Phi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus_x \Phi \mid \bigotimes_x \Phi \]

with \( a \in A \cup \{\triangleright\} \), \( d \in D \), \( s \in S \).
Challenging for the **Boolean part:**
we need unambiguous automata

Use deterministic automata
of size non-elementary...

Take advantage of the **pebbles**
to build **linear sized** automata

\[
\varphi ::= \top \mid P_a(x) \mid x = y \mid R_d(x, y) \mid R_d^+(x, y) \mid \neg \varphi \mid \varphi \lor \varphi \mid \exists x \varphi
\]

\[
\Phi ::= s \mid \varphi? \Phi : \Phi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus_x \Phi \mid \bigotimes_x \Phi
\]

with \( a \in A \cup \{\triangleright\}, \ d \in D, \ s \in S. \)
wFO to pebWA

\[ \varphi ::= \top \mid P_a(x) \mid x = y \mid R_d(x, y) \mid R_d^+(x, y) \mid \neg \varphi \mid \varphi \lor \varphi \mid \exists x \varphi \]

\[ \Phi ::= s \mid \varphi? \Phi : \Phi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus_x \Phi \mid \bigotimes_x \Phi \]

with \( a \in A \cup \{\triangleright\} \), \( d \in D \), \( s \in S \).

\[ \varphi = R_d(x, y) \]

\[ \varphi = R_d^+(x, y) \]
wFO to pebWA

\[ \varphi ::= \top \mid P_a(x) \mid x = y \mid R_d(x, y) \mid R_d^+(x, y) \mid \neg \varphi \mid \varphi \lor \varphi \mid \exists x \varphi \]

\[ \Phi ::= s \mid \varphi? \Phi : \Phi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus_x \Phi \mid \bigotimes_x \Phi \]

with \( a \in A \cup \{\triangleright\} \), \( d \in D \), \( s \in S \).

Disjunction/conjunction
\[ \xi = \varphi \lor \psi \]

Existential/Universal quantifications
\[ \xi = \exists x \varphi \]
### wFO to pebWA

\[ \varphi ::= \top \mid P_a(x) \mid x = y \mid R_d(x, y) \mid R_d^+(x, y) \mid \neg \varphi \mid \varphi \lor \psi \mid \exists x \varphi \]

\[ \Phi ::= s \mid \varphi? \Phi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus_x \Phi \mid \bigotimes_x \Phi \]

with \( a \in A \cup \{\rhd\} \), \( d \in D \), \( s \in S \).

**Disjunction/conjunction**

\[ \xi = \varphi \lor \psi \]

**Existential/Universal quantifications**

\[ \xi = \exists x \varphi \]

**Thm:** Consider a searchable class of graphs.

For each wFO formula, we can construct an equivalent layered pebWA,
- \# states linear in the size of the formula
- \# layers = quantifier depth of the formula
- \# pebble names = \# variable names in the formula
Model Checking
Query Evaluation
Complexity of evaluating queries

Given a layered pebWA with $p$ pebble names and a graph, we can compute $\llbracket A \rrbracket(G)$ with $O((p+1)|Q|^3|V|^{p+3})$ scalar operations (sum, product, star).

**Reusability:** the number of layers (quantifier depth in formulas) may be much bigger than the number of pebble names (x and y, e.g.)

Given a layered pebWA with $p$ pebble names and a (nested) word, we can compute $\llbracket A \rrbracket(W)$ with $O((p+1)|Q|^3|W|^{p+1})$ scalar operations (sum, product, star).

**strongly layered:** each layer is associated with a fixed pebble name: $O(|Q|^3|W|^p)$
Getting rid of variables
Weighted Expressions
Dynamic Logics (PDL)
Temporal Logics
Expressing more properties: path

In First-Order Logic, you cannot follow unbounded complex paths.

A path: \( \rightarrow \downarrow \downarrow \downarrow \rightarrow \rightarrow \uparrow \)

A pattern: \((\downarrow + \rightarrow)^* \cdot \uparrow \cdot (\downarrow + \leftarrow)^*\)

What is the length of the shortest path from \(\square\) to \(\blacksquare\)?

\(((\square ? \lor \square ?) \cdot 1 \cdot (\rightarrow + \leftarrow + \downarrow + \uparrow))^* \cdot \blacksquare?)

**Weighted Expressions** and **Hybrid Weighted Expressions**

navigating over graphs
Summary of Expressiveness

- Weighted Hybrid Regular Expressions
- Weighted Hybrid PDL
- Layered Weighted Automata with Pebbles
- Weighted Linear Temporal Logic
- Weighted First-Order Logic

Efficient translations: automata of linear size
Summary

Several ways of specifying quantitative properties of graphs: regular expressions with pebbles, weighted hybrid PDL, weighted temporal logic, weighted first-order logic.

Efficient translation of the queries into pebWA

Efficient evaluation (model-checking of database community)
Perspectives

Extend wFO with a bounded transitive closure operator in order to have the same expressive power as pebWA. (See next talk of Benjamin)

Add data or weights to the models (graphs)

Lift Model evaluation to System evaluation

Add comparisons of weights to the specification formalisms