Logics for Weighted Automata and Transducers

Benjamin Monmege
LIF, Aix-Marseille Université, France

Based on joint works with Paul Gastin, Benedikt Bollig and Marc Zeitoun
Software Verification
Software Verification

Critical Software
- communication systems
- e-commerce
- health databases
- energy production
Software Verification

Critical Software
- communication systems
- e-commerce
- health databases
- energy production
Software Verification

Property to be verified

Critical Software
- communication systems
- e-commerce
- health databases
- energy production
Software Verification

Critical Software

- communication systems
- e-commerce
- health databases
- energy production

Property to be verified

Is the property verified or not by the software?
Software Verification

- Critical Software
- Communication systems
- E-commerce
- Health databases
- Energy production

Property to be verified

- 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?
Software Verification

Property to be verified

- 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?
Formal Verification

- Property to be verified
- Critical Software:
  - communication systems
  - e-commerce
  - health databases
  - energy production

Is the property verified or not by the software?
Formal Verification

Property to be verified

Is the property verified or not by the model?

Formal Model

ababcaabb

Critical Software
- communication systems
- e-commerce
- health databases
- energy production

TO BE VERIFIED
Formal Verification

Property to be verified
Formal Specification

\[(a + b)^* c (ac)^+\]

\[\forall x \forall y (x < y \Rightarrow \exists z (x < z < y))\]

\[FG(p \cup q)\]

Is the property verified or not by the model?

Critical Software
- communication systems
- e-commerce
- health databases
- energy production

TO BE VERIFIED
Qualitative/Quantitative

- Qualitative, Boolean: [Büchi’60], [Elgot’61], [Trakhtenbrot’61]

Automata  MSO logic
Qualitative/Quantitative

- Qualitative, Boolean: [Büchi’60], [Elgot’61], [Trakhtenbrot’61]
- Quantitative, weights

![Diagram](image)
Qualitative/Quantitative

- Qualitative, Boolean: [Büchi’60], [Elgot’61], [Trakhtenbrot’61]

- Quantitative, weights

- Automata

- MSO logic

- Weighted Automata

- ???
Qualitative/Quantitative

- Qualitative, Boolean: [Büchi’60], [Elgot’61], [Trakhtenbrot’61]

- Quantitative, weights

Find suitable weighted MSO logic
Qualitative/Quantitative

- Qualitative, Boolean: [Büchi’60], [Elgot’61], [Trakhtenbrot’61]

Focus on definability / qualitative

Find suitable weighted MSO logic
Qualitative/Quantitative

- Qualitative, Boolean: [Büchi’60], [Elgot’61], [Trakhtenbrot’61]
- Quantitative, weights

Find suitable weighted MSO logic
Weighted Automata

\[ \Sigma, 1 \]

\[ \Sigma, 1 \]

\[ 0 \]

\[ 1 \]

\[ a, 1 \]

\[ b, -1 \]

\[ a, 0 \]

\[ b, 0 \]

\[ b, 0 \]

\[ 2 \]

\[ a, 1 \]

\[ b, 0 \]

\[ b, 0 \]
Weighted Automata

\[ (\mathbb{Z}, +, \times, 0, 1) \]
Weighted Automata

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

\[
\begin{array}{c}
\begin{array}{cc}
0 & \xrightarrow{a, 1} 1 \\
& \xrightarrow{b, -1} 1
\end{array} &
\begin{array}{cc}
\Sigma, 1 & \xrightarrow{a, 1} \Sigma, 1
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{cc}
0 & \xrightarrow{a, 0} 1 \\
b, 0 & \xrightarrow{b, 0} 2
\end{array} &
\begin{array}{cc}
1 & \xrightarrow{a, 1} 1 \\
b, 0 & \xrightarrow{b, 0} 2
\end{array}
\end{array}
\]
Weighted Automata

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

\[0 \xrightarrow{a, 1} 1 \xrightarrow{b, -1} 1 \xrightarrow{a, 1} 1\]

\[0 \xrightarrow{a, 1} 0 \xrightarrow{b, -1} 1 \xrightarrow{a, 1} 1\]
Weighted Automata

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

- From 0 to 1: \( a, 1 \) (weight 1)
- From 1 to 2: \( b, 0 \) (weight 0)
- From 2 to 0: \( a, 0 \) (weight 0)
- From 0 to 0: \( a, 1 \) (weight 1)
- From 0 to 1: \( b, -1 \) (weight -1)
- From 1 to 0: \( a, 1 \) (weight 1)
- From 1 to 1: \( a, 1 \) (weight 1)
- From 1 to 2: \( b, 0 \) (weight 0)
- From 2 to 2: \( a, 0 \) (weight 0)

The automaton transitions are labeled with inputs and weights.
Weighted Automata

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

Semantics of \( aba \): \( 1 + (-1) + 1 = 1 \)
Weighted Automata

\[
\begin{align*}
\Sigma, 1 & \quad \Sigma, 1 \\
0 & \xrightarrow{a, 1} 1 \xrightarrow{b, -1} 0 \xrightarrow{b, 0} 1 \xrightarrow{a, 1} 0 \\
0 & \xrightarrow{a, 1} 1 \xrightarrow{b, -1} 0 \xrightarrow{b, 0} 0 \xrightarrow{a, 1} 1 \\
0 & \xrightarrow{a, 1} 0 \xrightarrow{b, 1} 0 \xrightarrow{a, 1} 1 \\
\end{align*}
\]

Semantics of \textit{aba}: \( 1 + (-1) + 1 = 1 \)
Weighted Automata

Semantics of \( aba \): \( 1 + (-1) + 1 = 1 \)
Weighted Automata

\[ \Sigma, 1 \]

\[ \Sigma, 1 \]

0 \( \xrightarrow{a,1} \) 1 \( \xrightarrow{b,-1} \) 1 \( \xrightarrow{a,1} \) 1

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

Semantics of \( aba \): \( 1+(-1)+1 = 1 \)
Weighted Automata

\[
\Sigma, 1 \\
0 \xrightarrow{a, 1} 1 \xrightarrow{b, -1} 0
\]

\[
\Sigma, 1 \\
0 \xrightarrow{a, 1} 1 \xrightarrow{b, 0} 2
\]

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

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

Semantics of \(aba\): \(1 + (-1) + 1 = 1\)
**Weighted Automata**

\[\Sigma, 1\]

\[\Sigma, 1\]

\[\begin{array}{c}
0 \\
\ \ \ \ \ \ \ \ a, 1 \\
\ \ \ \ \ \ \ \ b, -1 \\
1
\end{array}\]

\[\begin{array}{c}
\Sigma, 1 \\
\ \ a, 0 \\
\ \ b, 0 \\
\ \ a, 1 \\
\ \ b, 0 \\
\ \ a, 0 \\
\ \ b, 0 \\
\ \ a, 0 \\
\end{array}\]

\[\#_a(w) - \#_b(w)\]

\[\mathbb{Z}, +, \times, 0, 1\]

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

Semantics of \textit{aba}: \(1 + (-1) + 1 = 1\)

Semantics of \textit{aaba}: \(\max(2, 1) = 2\)
Weighted Automata

Semantics of \( aba \): \( 1 + (-1) + 1 = 1 \)

Semantics of \( aaba \): \( \max(2, 1) = 2 \)
How to Specify Quantitative Properties?
How to Specify Quantitative Properties?

Weighted Monadic Second Order Logic [Droste&Gastin 05]
generalized to trees [Droste&Vogler 06], infinite words [Droste&Rahonis 07],
nested words [Mathissen 10] or pictures [Fichtner 11]
How to Specify Quantitative Properties?

Weighted Monadic Second Order Logic [Droste&Gastin 05] generalized to trees [Droste&Vogler 06], infinite words [Droste&Rahonis 07], nested words [Mathissen 10] or pictures [Fichtner 11]

Weighted Regular Expressions over finite words [Kleene 56, Schützenberger 61]
How to Specify Quantitative Properties?

**Weighted Monadic Second Order Logic** [Droste&Gastin 05] generalized to trees [Droste&Vogler 06], infinite words [Droste&Rahonis 07], nested words [Mathissen 10] or pictures [Fichtner 11]

**Weighted Regular Expressions** over finite words [Kleene 56, Schützenberger 61]

**Weighted Temporal Logics:**
PCTL [Hansson&Jonsson 94], WLTL [Mandrali 12]
How to Specify Quantitative Properties?

**Weighted Monadic Second Order Logic** [Droste&Gastin 05] generalized to trees [Droste&Vogler 06], infinite words [Droste&Rahonis 07], nested words [Mathissen 10] or pictures [Fichtner 11]

**Weighted Regular Expressions** over finite words [Kleene 56, Schützenberger 61]

**Weighted Temporal Logics:**
- PCTL [Hansson&Jonsson 94], WLTL [Mandrali 12]

- Core weighted logic for weighted automata
- Enhancing the logic to handle more properties: FO vs pebbles
- A special case: the transducers
Weighted MSO

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

[Droste-Gastin, 05]
Weighted MSO

\[ \varphi ::= s \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg P_a(x) \mid \neg(x \leq y) \mid \neg(x \in X) \]

\[ \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \exists x \varphi \mid \forall x \varphi \mid \exists X \varphi \mid \forall X \varphi \]

Negation restricted to atomic formulae

[Droste-Gastin, 05]
Weighted MSO

\( \varphi ::= s \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg P_a(x) \mid \neg (x \leq y) \mid \neg (x \in X) \)

\( \varphi \lor \varphi \mid \varphi \land \varphi \mid \exists x \varphi \mid \forall x \varphi \mid \exists X \varphi \mid \forall X \varphi \)

Arbitrary constants from a semiring

Negation restricted to atomic formulae

[Droste-Gastin, 05]
Weighted MSO

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

- Semantics in a semiring: \( \mathbb{S} = (S, +, \times, 0, 1) \)
- Atomic formulae: 0, 1
- disjunction, existential quantifications: sum
- conjunction, universal quantifications: product
- Inspired from the boolean semiring: \( \mathbb{B} = (\{0, 1\}, \lor, \land, 0, 1) \)

[Droste-Gastin, 05]
Weighted MSO

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

- Examples

\[ \varphi_1 = \exists x P_a(x) \]

\[ \llbracket \varphi_1 \rrbracket(w) = |w|_a \]

[Droste-Gastin, 05]
Weighted MSO

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

- **Examples**

\[ \varphi_1 = \exists x P_a(x) \]
\[ \llbracket \varphi_1 \rrbracket (w) = |w|_a \]

\[ \varphi_2 = \forall x \exists y (y \leq x \land P_a(y)) \]
\[ \llbracket \varphi_2 \rrbracket (abaab) = 1 \times 1 \times 2 \times 3 \times 3 \]
\[ \llbracket \varphi_2 \rrbracket (a^n) = n! \]

[Droste-Gastin, 05]
Weighted MSO

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

Examples

\[ \varphi_1 = \exists x P_a(x) \quad \varphi_2 = \forall x \exists y (y \leq x \land P_a(y)) \]
\[ \llbracket \varphi_1 \rrbracket (w) = |w|_a \quad \llbracket \varphi_2 \rrbracket (abaab) = 1 \times 1 \times 2 \times 3 \times 3 \]
\[ \llbracket \varphi_2 \rrbracket (a^n) = n! \]

Too big to be computed by a weighted automaton

[Droste-Gastin, 05]
Weighted MSO

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

**Examples**

\[ \phi_1 = \exists x \, P_a(x) \]
\[ \phi_2 = \forall x \, \exists y \, (y \leq x \land P_a(y)) \]

\[ [\phi_1](w) = \mid w \mid_a \]
\[ [\phi_2](abaab) = 1 \times 1 \times 2 \times 3 \times 3 \]
\[ [\phi_2](a^n) = n! \]

We need to restrict weighted MSO

Too big to be computed by a weighted automaton

[Droste-Gastin, 05]
Weighted MSO

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

Theorem: weighted automata = restricted wMSO

[Droste-Gastin, 05]
Weighted MSO

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

Theorem: weighted automata = restricted wMSO

[Droste-Gastin, 05]
Weighted MSO

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

\[ \varphi \text{ almost boolean} \]

Theorem: weighted automata = restricted wMSO

[Droste-Gastin, 05]
Weighted MSO

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

Theorem: weighted automata = restricted wMSO

[Droste-Gastin, 05]
Core weighted MSO logic

- Boolean fragment

\[ \varphi ::= \top | P_a(x) | x \leq y | x \in X | \neg \varphi | \varphi \land \varphi | \forall x \varphi | \forall X \varphi \]

[Gastin-Monmege, 15]
Core weighted MSO logic

- **Boolean fragment**

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

- **Step formulae**

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

[Gastin-Monmege, 15]
Core weighted MSO logic

- Boolean fragment

\[ \varphi ::= \top | P_a(x) | x \leq y | x \in X | \neg \varphi | \varphi \land \varphi | \forall x \varphi | \forall X \varphi \]

- Step formulae

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

if ... then ... else ...
Core weighted MSO logic

- Boolean fragment

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

- Step formulae

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

\[ P_a(x) ? 1 : 0 \]

[Gastin-Monmege, 15]
Core weighted MSO logic

- **Boolean fragment**

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

- **Step formulae**

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

\[ P_a(x) ? 1 : 0 \]

\[ P_a(x) ? 1 : (P_b(x) ? -1 : 0) \]

[Gastin-Monmege, 15]
Core weighted MSO logic

- **Boolean fragment**

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

- **Step formulae**

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

\[ P_a(x) ? 1 : 0 \]

\[ P_a(x) ? 1 : (P_b(x) ? -1 : 0) \]

\[ x \in X_1 ? s_1 : (x \in X_2 ? s_2 : \cdots (x \in X_{n-1} ? s_{n-1} : s_n) \cdots ) \]

[Gastin-Monmege, 15]
Core weighted MSO logic

- **Boolean fragment**

\[ \phi ::= \top \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg \phi \mid \phi \land \phi \mid \forall x \phi \mid \forall X \phi \]

- **Step formulae**

\[ \Psi ::= s \mid \phi \, ? \Psi : \Psi \]

\[
P_a(x) \, ? 1 : 0
\]

\[
P_a(x) \, ? 1 : (P_b(x) \, ? -1 : 0)
\]

\[
x \in X_1 \, ? s_1 : (x \in X_2 \, ? s_2 : \cdots (x \in X_{n-1} \, ? s_{n-1} : s_n) \cdots)
\]

\[
[\Psi](w, \sigma) = s
\]

*[Gastin-Monmege, 15]*
Core weighted MSO logic

- **Boolean fragment**

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

- **Step formulae**

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

A step formula takes finitely many values
For each value, the pre-image is MSO-definable

Use ... some value occurring in \( \Psi \)

[Gastin-Monmege, 15]
Core weighted MSO logic

- **Boolean fragment**
  \[ \varphi ::= \top \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg \varphi \mid \varphi \land \varphi \mid \forall x \varphi \mid \forall X \varphi \]

- **Step formulae**
  \[ \Psi ::= s \mid \varphi ? \Psi : \Psi \]

- **core wMSO**
  \[ \Phi ::= 0 \mid \varphi ? \Phi : \Phi \mid \Phi + \Phi \mid \sum_x \Phi \mid \sum_X \Phi \mid \prod_x \Psi \]

[Gastin-Monmege, 15]
Core weighted MSO logic

- **Boolean fragment**
  \[ \varphi ::= \top | P_a(x) | x \leq y | x \in X | \neg \varphi | \varphi \land \varphi | \forall x \varphi | \forall X \varphi \]

- **Step formulae**
  \[ \Psi ::= s | \varphi ? \Psi : \Psi \]

- **Core wMSO**
  \[ \Phi ::= 0 | \varphi ? \Phi : \Phi | \Phi + \Phi | \sum_x \Phi | \sum_X \Phi | \prod_x \Psi \]

[no constants]

[Gastin-Monmege, 15]
Core weighted MSO logic

- **Boolean fragment**
  \[ \varphi ::= \top \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg \varphi \mid \varphi \land \varphi \mid \forall x \varphi \mid \forall X \varphi \]

- **Step formulae**
  \[ \Psi ::= s \mid \varphi ? \Psi : \Psi \]

- **core wMSO**
  \[ \Phi ::= 0 \mid \varphi ? \Phi : \Phi \mid \Phi + \Phi \mid \sum_x \Phi \mid \sum_X \Phi \mid \prod_x \Psi \]

- **Core weighted MSO**
  - no constants
  - if ... then ... else ...

---

[Gastin-Monmege, 15]
Core weighted MSO logic

- **Boolean fragment**
  \[ \varphi ::= \top | P_a(x) | x \leq y | x \in X | \neg \varphi | \varphi \land \varphi | \forall x \varphi | \forall X \varphi \]

- **Step formulae**
  \[ \Psi ::= s | \varphi ? \Psi : \Psi \]

- **core wMSO**
  \[ \Phi ::= 0 | \varphi ? \Phi : \Phi | \Phi + \Phi | \sum_x \Phi | \sum_X \Phi | \prod_x \Psi \]

  no constants
  if ... then ... else ...

Assigns a value from \(\Psi\) to each position

[Gastin-Monmege, 15]
Core weighted MSO logic

- **Boolean fragment**
  \[ \varphi ::= \top | P_a(x) | x \leq y | x \in X | \neg \varphi | \varphi \land \varphi | \forall x \varphi | \forall X \varphi \]

- **Step formulae**
  \[ \Psi ::= s | \varphi ? \Psi : \Psi \]

- **core wMSO**
  \[ \Phi ::= 0 | \varphi ? \Phi : \Phi | \Phi + \Phi | \sum_x \Phi | \sum_X \Phi | \prod_x \Psi \]

  no constants

  if ... then ... else ...

  Assigns a value from \( \Psi \) to each position

  \[
  \{ \prod_x \Psi \}(w, \sigma) = \{\{([\Psi](w, \sigma[x \mapsto i]))_i\}_i\} \in \mathbb{N}\langle R^* \rangle
  \]

  [Gastin-Monmege, 15]

  singleton multiset
Core weighted MSO logic

- **Boolean fragment**

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

- **Step formulae**

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

- **core wMSO**

  \[ \Phi ::= 0 \mid \varphi ? \Phi : \Phi \mid \Phi + \Phi \mid \sum_x \Phi \mid \sum_X \Phi \mid \prod_x \Psi \]

- **Semantics**

  - \( \{ \mid 0 \mid \}(w, \sigma) = \emptyset \)
  
  - **sums over multisets**

    \[ \{ \mid \prod_x \Psi \mid \}(w, \sigma) = \{(\{[\Psi](w, \sigma[x \mapsto \text{\textit{i}}])\}_{\text{\textit{i}}} \in \mathbb{N}(R^*) \]

  [Gastin-Monmege, 15]
Multisets of weight structures

- A run generates a sequence of weights $wgt(\rho) = s_1s_2 \cdots s_n$
- Abstract semantics $\{\mathcal{A}\}(w) = \{wgt(\rho) | \rho \text{ run on } w\}$

[Gastin-Monmege, 15][Droste-Perevoshchikov, 14]
Multisets of weight structures

- A run generates a sequence of weights $\text{wgt}(\rho) = s_1 s_2 \cdots s_n$

- Abstract semantics $\{ |A| \}(w) = \{ \{ \text{wgt}(\rho) \mid \rho \text{ run on } w \} \}$

$\{ |A| \} : \Sigma^* \to \mathbb{N}\langle R^* \rangle$

[Gastin-Monmege, 15][Droste-Perevoshchikov, 14]
Multisets of weight structures

- A run generates a sequence of weights \( \text{wgt}(\rho) = s_1 s_2 \cdots s_n \)
- Abstract semantics  
  \( \{|A|\}(w) = \{\text{wgt}(\rho) \mid \rho \text{ run on } w\} \)

\[\{|A|\} : \Sigma^* \rightarrow \mathbb{N}\langle R^* \rangle\]

- Aggregation  
  \( \text{aggr} : \mathbb{N}\langle R^* \rangle \rightarrow S \)

[Gastin-Monmege, 15][Droste-Perevoshchikov, 14]
Multisets of weight structures

Semiring: sum-product

\[ \text{aggr}_{sp}(A) = \sum \prod A = \sum_{r_1 \cdots r_n \in A} r_1 \times \cdots \times r_n \]

\[ \{\mathcal{A}\} : \Sigma^* \to \mathbb{N}\langle R^*\rangle \]

- Aggregation

\[ \text{aggr} : \mathbb{N}\langle R^*\rangle \to S \]

[Gastin-Monmege, 15][Droste-Perevoshchikov, 14]
Multisets of weight structures

Semiring: sum-product
\[ \text{aggr}_{sp}(A) = \sum \prod A = \sum_{r_1 \cdots r_n \in A} r_1 \times \cdots \times r_n \]

Valuation monoid: sum-valuation
\[ \text{aggr}_{sv}(A) = \sum \text{Val}(A) = \sum_{r_1 \cdots r_n \in A} \text{Val}(r_1 \cdots r_n) \]

- Aggregation
\[ \text{aggr}: \mathbb{N} \langle R^* \rangle \rightarrow S \]

[Gastin-Monmege, 15][Droste-Perevoshchikov, 14]
Multisets of weight structures

**Semiring: sum-product**

\[ \text{aggr}_{sp}(A) = \sum \prod A = \sum_{r_1 \cdots r_n \in A} r_1 \times \cdots \times r_n \]

**Valuation monoid: sum-valuation**

\[ \text{aggr}_{sv}(A) = \sum \text{Val}(A) = \sum_{r_1 \cdots r_n \in A} \text{Val}(r_1 \cdots r_n) \]

- Aggregation

\[ \rightarrow S \]

[Gastin-Monmege, 15][Droste-Perevoshchikov, 14]
Multisets of weight structures

- A run generates a sequence of weights \( \text{wgt}(\rho) = s_1 s_2 \cdots s_n \)

- Abstract semantics \( \{\mathcal{A}\}(w) = \{\{\text{wgt}(\rho) \mid \rho \text{ run on } w\}\} \)

- Aggregation \( \text{aggr} : \mathbb{N}\langle R^* \rangle \rightarrow S \)

- Concrete semantics \( [\mathcal{A}] = \text{aggr} \circ \{\mathcal{A}\} : \Sigma^* \rightarrow S \)

[Gaslin-Monmege, 15][Droste-Perevoshchikov, 14]
Core weighted MSO logic

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

\[ \psi ::= s \mid \varphi ? \psi : \psi \]

\[ \Phi ::= 0 \mid \varphi ? \Phi : \Phi \mid \Phi + \Phi \mid \sum_x \Phi \mid \sum_X \Phi \mid \prod_x \psi \]

Theorem: weighted automata = core wMSO

[Gastin-Monmege, 15]
Core weighted MSO logic

\[ \varphi ::= \top | P_a(x) | x \leq y | x \in X | \neg \varphi | \varphi \land \varphi | \forall x \varphi | \forall X \varphi \]

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

\[ \Phi ::= 0 | \varphi \Downarrow \Phi : \Phi | \Phi + \Phi | \sum_x \Phi | \sum_X \Phi | \prod_x \Psi \]

Theorem: weighted automata = core wMSO

- Abstract semantics
  \[ \{ - \} : \Sigma^* \rightarrow \mathbb{N}\langle R^*\rangle \]

[Gastin-Monmege, 15]
Core weighted MSO logic

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

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

\[ \Phi ::= 0 \mid \varphi ? \Phi : \Phi \mid \Phi + \Phi \mid \sum_x \Phi \mid \sum_X \Phi \mid \prod_x \Psi \]

Theorem: weighted automata = core wMSO

- Abstract semantics \[ \{ - \} : \Sigma^* \rightarrow \mathbb{N}\langle R^* \rangle \]
- Concrete semantics \[ [-] = \text{aggr} \circ \{ - \} : \Sigma^* \rightarrow S \]

[Gastin-Monmege, 15]
Core weighted MSO logic

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

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

\[ \Phi ::= 0 \mid \varphi \ ? \Phi : \Phi \mid \Phi + \Phi \mid \sum_x \Phi \mid \sum_X \Phi \mid \prod_x \Psi \]

Theorem: weighted automata = core wMSO

- Abstract semantics
- Concrete semantics

\[ \llbracket \cdot \rrbracket = \text{aggr} \circ \{ \cdot \} : \Sigma^* \rightarrow N \langle R^* \rangle \]

- Easy constructive proofs
- Preservation of the constants
- No restriction on core wMSO
- No hypotheses on weights

[Gastin-Monmege, 15]
Extensions

More general models than words: trees, nested words…

More powerful logics: deciding if a wMSO formula is expressible in core wMSO?

More powerful automata: finding equivalent fragments of wMSO
Weighted FO logic

\[ \varphi ::= \top | P_a(x) | x \leq y | x \in X | \neg \varphi | \varphi \land \varphi | \forall x \varphi | \forall X \varphi \]

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

\[ \Phi ::= 0 | \varphi ? \Phi : \Phi | \Phi + \Phi | \sum_x \Phi | \sum_X \Phi | \prod_x \Psi \]
Weighted FO logic

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

\[ \Phi ::= s \mid \varphi \cdot \Phi \mid \Phi + \Phi \mid \Phi \times \Phi \mid \sum_x \Phi \mid \prod_x \Phi \]

We can keep Boolean MSO or restrict to FO…

- Reintroduction of the product
- Unconditional product quantification
Weighted FO logic

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

\[ \Phi ::= s \mid \varphi ? \Phi \mid \Phi + \Phi \mid \Phi \times \Phi \mid \sum_x \Phi \mid \prod_x \Phi \]

We can keep Boolean MSO or restrict to FO…

Reintroduction of the product

Unconditional product quantification

\[ \varphi_2 = \forall x \exists y \ (y \leq x \land P_a(y)) \quad \text{with } \lbrack \varphi_2 \rbrack(a^n) = n! \]
Weighted FO logic

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

\[ \Phi ::= s \mid \varphi \Phi \mid \Phi + \Phi \mid \Phi \times \Phi \mid \sum_x \Phi \mid \prod_x \Phi \]

We can keep Boolean MSO or restrict to FO...

Reintroduction of the product

Unconditional product quantification

\[ \left[ \prod_x \prod_y 2 \right](w) = 2^{|w|^2} \]
Pebble weighted automata

\[ A = (Q, A, I, \delta, T) \]

\[ I \in S^Q \quad T \in S^Q \]

\[ \delta: Q \times \text{Test} \times \text{Move} \times Q \to S \]

\[ \text{Move} = \{ \rightarrow, \leftarrow, \uparrow \} \cup \{ \downarrow_x \mid x \in \text{Peb} \} \]

Run as a finite sequence of configurations \((W, \sigma, q, i, \pi)\)

with free pebbles \(\sigma: \text{Peb} \to \text{pos}(W)\)

and a stack of currently dropped pebbles \(\pi \in (\text{Peb} \times \text{pos}(W))^*\)
Text: Weighted automata with pebbles

Transition: $q \xrightarrow{\text{weight test move}} q$ 0

Run $\rightarrow$ over word $u \in A^+$: sequence of transitions.

weight $(\rightarrow)$: product of weights of transitions

Non determinism: $[A_{p,q}(u)] = \sum_{\rightarrow} \text{run over } u$ weight $(\rightarrow)$

References:

[Globerman and Harel, 96][Engelfriet and Hoogeboom, 99]
Pebble weighted automata

\[ \text{Weighted automata with pebbles} \]

\[ \text{Move} = \{ 1, 2, !, " \} \]

\[ \text{Transition: } q \xrightarrow{\text{weight test move}} q_0 \]

\[ \text{Run } \mapsto \text{over word } u \in A^+ \text{: sequence of transitions.} \]

\[ \text{weight} (\mapsto) = \text{product of weights of transitions} \]

\[ \text{Non determinism: } [\[ A \, p,q \]] (u) = \text{run over } u \text{ weight} (\mapsto) \]

\[ ▹ a \ x b \ x b \ x c \ x a \ x c \ x b \ x a ▼ \]

\[ 5 \times 1 \times 7 \times 7 \]

\[ [\text{Globerman and Harel, 96}] [\text{Engelfriet and Hoogeboom, 99}] \]
Pebble weighted automata

Move = \{ [\!]_1, [\!]_2, "x" \} 

Pebbles:

\[ x | x | x^2 \]

Non determinism:
\[ [\!]_A \cdot [\!]_q (u) = \bigcup \{ \rightarrow \text{run over } u \} \]

weight \( (\rightarrow \text{run}) \): product of weights of transitions

Run \( \rightarrow \text{over word } u \) : sequence of transitions.

A\rightarrow
\begin{align*}
\text{a?} &\rightarrow 7b? \rightarrow \\
5 \rightarrow & \rightarrow \\
\triangleup? \rightarrow & \rightarrow \\
c? \downarrow & \rightarrow \\
\end{align*}

\begin{align*}
\text{2} \rightarrow & x? \rightarrow \\
3 \rightarrow & x? \rightarrow \\
3 \rightarrow & \rightarrow ? \leftarrow \\
\end{align*}

\begin{align*}
5 \times 1 \times 7 \times 7 \times 1
\end{align*}

[Globerman and Harel, 96][Engelfriet and Hoogeboom, 99]
Pebble weighted automata

\[
\begin{align*}
\text{Move} & = \{ \!, \, \cdot \, \} \\
\text{Transition:} & \quad q \xrightarrow{weight \, test \, move} ! q
\end{align*}
\]

Run \( \rightarrow \) over word \( u \in A^+ \): sequence of transitions.

\[
\text{weight} (\rightarrow) : \text{product of weights of transitions}
\]

Non determinism:
\[
\left[ \begin{array}{c}
A \\
\end{array} \right] (u) = \bigcup_{\text{run over } u} \text{weight} (\rightarrow)
\]

\[Globerman \text{ and Harel, 96}\][Engelfriet and Hoogeboom, 99]
Pebble weighted automata

\[ a? \rightarrow 7b? \rightarrow \]
\[ 5 \leftarrow \]
\[ c? \downarrow x \]
\[ 2 \leftarrow x? \rightarrow \]
\[ 3 \leftarrow \]
\[ 3 \leftarrow \]

\[ 5 \times 1 \times 7 \times 7 \times 1 \]
\[ x \times 2 \times 2 \times 2 \times 1 \]

[Globerman and Harel, 96][Engelfriet and Hoogeboom, 99]
Pebble weighted automata

\[ \text{Move} = \{, !, " \}\]

\[ \text{Transition: } q \text{ weight test move } ! q \]

Run \( \Rightarrow \) over word: sequence of transitions.

weight (\( \Rightarrow \)) : product of weights of transitions

Non determinism:

\[ [\langle A \rangle_{p,q}] (u) = X \Rightarrow \text{run over } u \text{ weight (} \Rightarrow \) ) \]

\[ \begin{array}{c}
\text{Globerman and Harel, 96} \\
\text{Engelfriet and Hoogeboom, 99}
\end{array} \]
Pebble weighted automata

\[ \text{Run} \xrightarrow{\cdot} \text{over word } u \subseteq A^+ : \text{sequence of transitions.} \]

\[ \text{weight}(\cdot) : \text{product of weights of transitions} \]

\[ \text{Non determinism: } [A_{p,q}] (u) = \bigcup_{\rightarrow \text{run over } u} \text{weight}(\cdot) \]

\[ = 5000 \]

\[ M = 0 \]

\[ B \quad \text{Globerman and Harel, 96}\]

\[ \text{Engelfriet and Hoogeboom, 99} \]
Pebble weighted automata

\[ \text{Move} = \{ \text{!, "} \} \]
\[ \text{Transition: } q \xrightarrow{\text{weight test move}} q_0 \]
\[ \text{Run } \rightarrow \text{over word } u \in A^+ : \text{sequence of transitions.} \]
\[ \text{weight } (\rightarrow) : \text{product of weights of transitions} \]

Non determinism:
\[ [A]_{p,q} (u) = \bigcup \text{\textbf{\textbullet}} \rightarrow \text{run over } u \text{ weight } (\rightarrow) \]

\begin{align*}
\text{Globerman and Harel, 96} & \quad \text{Engelfriet and Hoogeboom, 99}
\end{align*}
Pebble weighted automata

\[\text{Move} = \{,!,"\}\right] \begin{array}{|c|c|c|}
\hline
#x | x^2 & \text{Peb} \\
\hline
\end{array}\]

Transition: \(q \xrightarrow{\text{weight test move}} !q\)

Run \(\xrightarrow{\text{over word}} u \in \mathcal{A}^+\) : sequence of transitions.

weight \((\xrightarrow{\text{}})\) : product of weights of transitions

Non determinism:
\[
[A_p,q](u) = \bigcup \{ \xrightarrow{\text{run over } u} \text{weight } (\xrightarrow{\text{}}) \}
\]

\[\begin{align*}
5 \ x | x 7 x 7 x 1 \\
x 2 x 2 x 2 x 1 | x 3 x 1 \\
x | x | 1 \\
\end{align*}\]

\[\text{Globerman and Harel, 96}]\text{Engelfriet and Hoogeboom, 99}\]
Pebble weighted automata

Transition: $q_0 \xrightarrow{w_1} q_0$

Run: $\rightarrow$ over word $u \in A^+$: sequence of transitions.

Weight: $(\rightarrow)$: product of weights of transitions

Non determinism:\n$\Delta_{A, q, p}(\cdot) = X \rightarrow$ run over $u$, weight $(\rightarrow)$

\[ M = 0 \quad B = \begin{cases} a \mapsto 7 & \text{if } b \mapsto 1 \\ b \mapsto 2 & \text{if } c \mapsto 0 \\ c \mapsto 0 & \text{if } a \mapsto 1 \end{cases} \]

\[ x = \begin{array}{c} 2 \quad 3 \quad 4 \quad 5 \\ x_1 \quad x_2 \quad x_3 \quad x_2 \end{array} \]

\[[\text{Globerman and Harel, 96}][\text{Engelfriet and Hoogeboom, 99}]\]
Pebble weighted automata

\[ \text{Move} = \{ \text{!}, \text{"} \} \]

\[ \text{Transition: } q \xrightarrow{\text{weight test move}} ! q \]

Run \( \xrightarrow{\cdot} \) over word \( u \in A^* + \): sequence of transitions.

weight \( (\cdot) \): product of weights of transitions

Non determinism:

\[ [A]_{p,q} (u) = \bigcup \{ \cdot \xrightarrow{\cdot} \text{run over } u \} \text{weight} \]

\[ \begin{array}{c}
\text{Globerman and Harel, 96} \\
\text{Engelfriet and Hoogeboom, 99}
\end{array} \]
Pebble weighted automata

Weighted automata with pebbles

I Move = 

I Transition: q weight test move 

I Run \( \rightarrow \) over word u \( \in \) A

I weight (\( \rightarrow \)) : product of weights of transitions

Non determinism: \( \left[ \right. \) A p,q \( \left. \right] \) (u) = \( \times \)

\( \rightarrow \) run over u

I weight (\( \rightarrow \))

[50x382]\( \uparrow \)

5 \( \times \) 1 \( \times \) 7 \( \times \) 7 \( \times \) 1

x2 \( \times \) 2 \( \times \) 2 \( \times \) 2 \( \times \) 1 \( \times \) 3 \( \times \) 1

x1 \( \times \) x1

x2 \( \times \) 2 \( \times \) 2 \( \times \) 2 \( \times \) 2 \( \times \) 1 \( \times \) 3 \( \times \) 3 \( \times \) 3

[Globerman and Harel, 96][Engelfriet and Hoogeboom,99]
Weighted automata with pebbles

Move = \{ 2, 4, 6 \}

\[ \{ \#x | x \} \]

Transition:

$$\Delta q \text{ weight test move } q_0$$

Run $\rightarrow$ over word $u \in \mathcal{A}$: sequence of transitions.

$\text{weight} (\rightarrow)$: product of weights of transitions

Non determinism:

\[
\Delta [A, q, p] (u) = \cap \Delta \rightarrow \text{run over } u \text{ weight } (\rightarrow)
\]

$\xrightarrow{5} a \xrightarrow{7b} \xrightarrow{3} c \xrightarrow{1} a \xrightarrow{b} b \xrightarrow{c} a \xrightarrow{c} b \xrightarrow{a}$

$5 \times 1 \times 7 \times 7 \times 1$

$x2x2x2x1 \times 3x1 \times 1 \times 1 \times 1$

$x2x2x2x2x2x1 \times 3x3x3x3x1 \times 1 \times 7 \times 1$

\[\text{Globerman and Harel, 96}\][Engelfriet and Hoogeboom, 99]
Pebble weighted automata

\[ \text{I } \text{Move} = \{ a, b, c \} \]\[ \text{I } \text{Transition: } q \text{ weight test move } q_0 \]

\[ \text{I } \text{Run } \rhd \text{ over word } u \]

\[ \text{I } \text{weight } (\rhd) : \text{ product of weights of transitions} \]

Non determinism:
\[ \left[ A_{p,q} (u) \right] = X \rhd \text{run over } u \text{ weight } (\rhd) \]

\[ 1 \Rightarrow a \Rightarrow 7b \Rightarrow \]

\[ 5 \Rightarrow \triangleleft \Rightarrow \]

\[ c \Rightarrow x \]

\[ 2 \Rightarrow x \Rightarrow \]

\[ 3 \Rightarrow ? \Rightarrow \]

\[ x \Rightarrow 3 \Rightarrow \Rightarrow \]

\[ 5 \times \times 7 \times 7 \times 1 \]

\[ x2 \times 2 \times 2 \times 1 \times 3 \times 1 \]

\[ x \times 1 \times 1 \]

\[ x2 \times 2 \times 2 \times 2 \times 2 \times 1 \times 3 \times 3 \times 3 \times 3 \times 1 \]

\[ x \times 1 \times 7 \times 1 \]

Weight of the run: 35 562 240

[Globerman and Harel, 96][Engelfriet and Hoogeboom, 99]
Pebble weighted automata

Non deterministic

Non determinism resolved by sum

Weight of the run: 35 562 240

Globerman and Harel, 96][Engelfriet and Hoogeboom, 99]
Pebble weighted automata

Non determinism

Non determinism resolved by sum

Weight of the run: 35 562 240

Sum of the weights of the runs:

\[ 5.7^{\lceil \frac{|w|_b} \rceil} \prod_{w_i = c_i} 2^{i-2}(3^i - 1) \]

[Globerman and Harel, 96][Engelfriet and Hoogeboom, 99]
Translating a formula into an automaton

**Sum** by disjoint union of automata

**Product:**

**Sum quantification:**

**Product quantification:**
Translating a formula into an automaton

Challenging for the *Boolean part*:

need unambiguous automata
Translating a formula into an automaton
Challenging for the Boolean part:
need unambiguous automata

Use deterministic automata of size non-elementary...
Translating a formula into an automaton

Challenging for the *Boolean part*: need unambiguous automata

- Use deterministic automata of size *non-elementary*...
- Take advantage of the *pebbles* to build a *linear size* automaton
Challenging for the *Boolean part*: need unambiguous automata

Use deterministic automata of size non-elementary...

Take advantage of the *pebbles* to build a *linear size* automaton

Disjunction/conjunction \( \xi = \varphi \lor \psi \)
Translating a formula into an automaton

Challenging for the Boolean part:
need unambiguous automata

Use deterministic automata of size non-elementary...

Take advantage of the pebbles to build a linear size automaton

Disjunction/conjunction
\( \xi = \varphi \lor \psi \)

Existential/Universal quantifications
\( \xi = \exists x \varphi \)

Translating a formula into an automaton
Logic equivalent to PWA?

- Weighted FO misses a counting capability…

- Solution: weighted transitive closure operation
Logic equivalent to PWA?

- Weighted FO misses a counting capability…

- Solution: weighted transitive closure operation

\[ \varphi^1(x, y) = \varphi(x, y) \]

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

• Weighted FO misses a counting capability...

• Solution: weighted transitive closure operation

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

\[ \text{TC}_{xy} \varphi = \bigvee_{n \geq 1} \varphi^n \]
Logic equivalent to PWA?

• Weighted FO misses a counting capability…

• Solution: weighted transitive closure operation

\[
\varphi^1(x, y) = \varphi(x, y)
\]

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

\[
\text{TC}_{xy} \varphi = \bigvee_{n \geq 1} \varphi^n
\]

Bounded transitive closure : 
\[
N \cdot \text{TC}_{xy} \varphi = \text{TC}_{xy}(x - N \leq y \leq x + N \land \varphi)
\]
Logic equivalent to PWA?

- Weighted FO misses a counting capability…

- Solution: weighted transitive closure operation

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

**Theorem:** PWA = wFO + bounded-TC
Application to transductions

input word \rightarrow T \rightarrow output word
Application to transductions

input word \rightarrow T \rightarrow output word

Pattern matching/replacement
Application to transductions

input word \( T \) output word

Pattern matching/replacement

Tree/Graph rewriting
Application to transductions

- Pattern matching/replacement
- Tree/Graph rewriting
- Update of XML databases
Existing models over words

Functions

- Two-way Deterministic Finite-State Transducers
- Functional One-way Finite-State Transducers
- MSOT (à la Courcelle)
- Copyless Streaming String Transducers (Alur et al)
Existing models over words

Functions

- Two-way Deterministic Finite-State Transducers
- Functional One-way Finite-State Transducers
- MSOT (à la Courcelle)
- Copyless Streaming String Transducers (Alur et al)

Relations

- Two-way Non-Deterministic Finite-State Transducers
- Non-Deterministic Finite-State Transducers
- Non-deterministic Copyless Streaming String Transducers (Alur et Deshmukh)
- NMSOT (with free second-order variables)
Existing models over words

**Functions**
- Two-way Deterministic Finite-State Transducers
- Functional One-way Finite-State Transducers
- MSOT (à la Courcelle)
- Copyless Streaming String Transducers (Alur et al)

**Relations**
- Two-way Non-Deterministic Finite-State Transducers
- Non-Deterministic Finite-State Transducers
- Non-deterministic Copyless Streaming String Transducers (Alur et Deshmukh)
- NMSOT (with free second-order variables)

only finite valued relations…
Transduction as weights

• Desire: weight transitions with words… Difficult to equip A* with a semiring structure: how to combine several accepting runs?

• Works for deterministic or unambiguous automata: functional transducers

• For relations: semiring of languages

\[(2^{A^*}, \cup, \cdot, \emptyset, \{\varepsilon\})\]
Examples

$$\prod_x (P_x(a)?\{aa\} : (P_x(b)?\{bb\} : \emptyset))$$
Examples

\[ \Pi_x (P_x(a)?\{aa\} : (P_x(b)?\{bb\} : \emptyset)) \]

\[ aba \rightarrow aabbaa \]
Examples

\(\prod_x (\mathcal{P}(a)?\{aa\} : (\mathcal{P}(b)?\{bb\} : \emptyset))\)  
\(aba \rightarrow aabbaa\)

\(\prod_x (\mathcal{P}(\ast)?\{\text{insert}\} : (\mathcal{P}(a)?\{a\} : (\mathcal{P}(b) : \{b\})))\)
Examples

$$\prod_{x} (P(x)?\{aa\} : (P(x)?\{bb\} : \emptyset)) \quad aba \rightarrow aabbaa$$

$$\prod_{x} (P(x)?\{insert\} : (P(x)?\{a\} : (P(x)?\{b\}))) \quad a^*b \rightarrow ainsertb$$
Examples

$$\prod_x (P(x)?{aa} : ( P(x)?{bb} : \emptyset )) \quad aba \rightarrow aabbaa$$

$$\prod_x (P(\star)?{\text{insert}} : (P(x)?{a} : (P(x)?{b})))) \quad a^*b \rightarrow \text{ainsertb}$$

$$\sum_y P(y)(\star)?[\prod_x (x = y)?{\text{insert}} : (P(x)?{\varepsilon} : (P(x)?{a} : (P(x)?{b}))))]$$
Examples

$$\prod_x (P(x) ? \{aa\} : (P(x) ? \{bb\} : \emptyset)) \quad \text{aba} \rightarrow \text{aabbaa}$$

$$\prod_x (P(\star) ? \{\text{insert}\} : (P(x) ? \{a\} : (P(x) ? \{b\})))) \quad \text{a*b} \rightarrow \text{ainsertb}$$

$$\sum_y P(y) ? \prod_x (x = y) ? \{\text{insert}\} : (P(\star) ? \{\varepsilon\} : (P(x) ? \{a\} : (P(x) ? \{b\})))) \quad \text{a*b*a} \rightarrow \{\text{ainsertba, abinserta}\}$$
Examples

\[ \prod_x P_x(a)^{\{a\}} : (P_x(b)^{\{\varepsilon\}}) \times \prod_x P_x(a)^{\{\varepsilon\}} : (P_x(b)^{\{c\}}) \]
Examples

\[
\prod_x P_x(a)\{a\} : (P_x(b)\{\varepsilon\}) \times \prod_x P_x(a)\{\varepsilon\} : (P_x(b) : \{c\})
\]

ababbaabb \rightarrow aaaaaccccc
Examples

\[
\prod_x P_x(a)\{a\} : (P_x(b)\{\varepsilon\}) \times \prod_x P_x(a)\{\varepsilon\} : (P_x(b)\{c\})
\]

ababbaabb \rightarrow aaaaaccccc

Not comp. by 1-way Func Transducer
Examples

\[ \prod_x P_x (a) ? \{a\} : (P_x (b) ? \{\varepsilon\}) \times \prod_x P_x (a) ? \{\varepsilon\} : (P_x (b) ? \{\varepsilon\}) \]

Not comp. by 1-way Func Transducer

\[ ababbaabb \rightarrow aaaaccccc \]

\[ \prod_x P_x (a) ? \{a\} : (P_x (b) ? \{\varepsilon\}) + \prod_x P_x (a) ? \{\varepsilon\} : (P_x (b) \times \{c\}) \]
Examples

\[ \prod_x P_x(a)?\{a\} : (P_x(b)?\{\varepsilon\}) \times \prod_x P_x(a)?\{\varepsilon\} : (P_x(b)?\{\varepsilon\}) \]

Not comp. by 1-way Func Transducer

\[ ababbaaabb \rightarrow \text{aaaaccccc} \]

\[ \prod_x P_x(a)?\{a\} : (P_x(b)?\{\varepsilon\}) + \prod_x P_x(a)?\{\varepsilon\} : (P_x(b) \times \{c\}) \]

\[ ababbaaabb \rightarrow \{aaaa,cccccc\} \]
Examples

\[ \prod_x P_x(a)?\{a\} : (P_x(b)?\{\varepsilon\}) \times \prod_x P_x(a)?\{\varepsilon\} : (P_x(b)?\{\varepsilon\}) \]

\[ \text{Not comp. by 1-way Func Transducer} \]

\[ ababbaabb \rightarrow aaaa,cccccc \]

\[ \prod_x P_x(a)?\{a\} : (P_x(b)?\{\varepsilon\}) + \prod_x P_x(a)?\{\varepsilon\} : (P_x(b) \times \{c\}) \]

\[ ababbaabb \rightarrow \{aaaa,cccccc\} \]

\[ \prod_x P_x(a)?\{a,\varepsilon\} : (P_x(b)?\{b,\varepsilon\}) \]
Examples

\( \prod_{x} P_x(a) ? \{a\} : (P_x(b) ? \{\varepsilon\}) \times \prod_{x} P_x(a) ? \{\varepsilon\} : (P_x(b) ? \{\varepsilon\}) \)  

\( ababbaabb \rightarrow \text{aaaaccccc} \)

\( \prod_{x} P_x(a) ? \{a\} : (P_x(b) ? \{\varepsilon\}) + \prod_{x} P_x(a) ? \{\varepsilon\} : (P_x(b) \times \{c\}) \)  

\( ababbaabb \rightarrow \{aaaa, ccccc\} \)

\( \prod_{x} P_x(a) ? \{a, \varepsilon\} : (P_x(b) ? \{b, \varepsilon\}) \)  

\( aba \rightarrow \{\varepsilon,a,b,ab,ba,aa,aba\} \)
Examples

\[ \prod_x P_x(a)?\{a\} : (P_x(b)?\{\varepsilon\}) \times \prod_x P_x(a)?\{\varepsilon\} : (P_x(b)?\{\varepsilon\}) \]

\[ ababbaabbb \rightarrow \text{aaaacccccc} \]

\[ \prod_x P_x(a)?\{a\} : (P_x(b)?\{\varepsilon\}) + \prod_x P_x(a)?\{\varepsilon\} : (P_x(b) \times \{c\}) \]

\[ ababbaabbb \rightarrow \{\text{aaaa,ccccccc}\} \]

\[ \prod_x P_x(a)?\{a,\varepsilon\} : (P_x(b)?\{b,\varepsilon\}) \]

\[ aba \rightarrow \{\varepsilon,a,b,ab,ba,aa,aba\} \]

\[ \prod_x P_x(a)? A^* a A^* : (P_x(b)? A^* b A^*) \]
Examples

\[ \prod_x P_x(a) : \{a\} : (P_x(b) : \{\varepsilon\}) \times \prod_x P_x(a) : \{\varepsilon\} : (P_x(b) : \{\varepsilon\}) \]

ababbaaabb \rightarrow \text{aaaaccccc}

\[ \prod_x P_x(a) : \{a\} : (P_x(b) : \{\varepsilon\}) + \prod_x P_x(a) : \{\varepsilon\} : (P_x(b) \times \{c\}) \]

ababbaaabb \rightarrow \{aaaa,cccccc\}

\[ \prod_x P_x(a) : \{a, \varepsilon\} : (P_x(b) : \{b, \varepsilon\}) \]

aba

\[ \prod_x P_x(a) : A^* a A^* : (P_x(b) : A^* b A^*) \]

aba \rightarrow A^*aA^*bA^*aA^*
Transducers

\[ \prod_{x} P_x(a)? A^* a A^* : (P_x(b)? A^* b A^*) \]

Infinite-valued, but deterministic
Reverse?

$A | \varepsilon, \rightarrow$

$last?$

$a | a, \leftarrow$

$first?$

$b | b, \leftarrow$
Reverse?

Impossible in FO...  
... because of order of interpretation of product
Reverse?

Impossible in FO…
… because of order of interpretation of product

Solution: in this non-commutative setting, add right-to-left products
Reverse?

Impossible in FO... ... because of order of interpretation of product

Solution: in this non-commutative setting, add right-to-left products

\[ \bigcup_{x} \left( P_{x}(a)?\{a\} : (P_{x}(b)?\{b\}) \right) \]
Transitive closure

\( \varphi ::= \top \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg \varphi \mid \varphi \land \varphi \mid \forall x \varphi \)

\( \Phi ::= L \mid \varphi \cdot \Phi : \Phi \mid \Phi + \Phi \mid \Phi \times \Phi \mid \sum_x \Phi \mid \prod_x \Phi \mid N-TC_{x,y} \Phi \)
Transitive closure

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

\[ \Phi ::= L \mid \varphi \cdot \Phi : \Phi \mid \Phi + \Phi \mid \Phi \times \Phi \mid \sum_x \Phi \mid \prod_x \Phi \mid N-TC_{x,y} \]

Regular language
Transitive closure

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

\[ \Phi ::= L \mid \varphi \mathbin{?}\Phi : \Phi \mid \Phi + \Phi \mid \Phi \times \Phi \mid \sum_x \Phi \mid \prod_x \Phi \mid N\text{-}TC_{x,y} \Phi \]

Regular language

Able to define right-to-left product

\[ \prod_x \Phi(x) := [1\text{-}TC_{x,y}(y = x - 1?\Phi(x))]\text{(last, first)} \times \Phi\text{(first)} \]
Transitive closure

\( \varphi ::= \top \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg \varphi \mid \varphi \land \varphi \mid \forall x \varphi \)

\( \Phi ::= L \mid \varphi \cdot \Phi : \Phi \mid \Phi + \Phi \mid \Phi \times \Phi \mid \sum_x \Phi \mid \prod_x \Phi \mid N-TC_{x,y} \Phi \)

**Regular language**

- Able to define right-to-left product

\[ \prod_x \Phi(x) := [1-TC_{x,y}(y = x - 1?\Phi(x))](last, first) \times \Phi(first) \]

**Theorem:** Pebble Transducers = FO + bounded-TC

- with regular language productions

**linear** transformation from logic to transducers
Algorithmic questions
Algorithmic questions

Theorem: Evaluation of FO + bounded-TC with complexity $O(|\text{formula}| \times |\text{input}| \times \#\text{variables})$
Algorithmic questions

Theorem: Evaluation of FO + bounded-TC with complexity $O(|\text{formula}| \times |\text{input}| \times \#\text{variables})$

- In functional setting: it is decidable if a 2-way transducer is recognisable by a 1-way transducer [Filiot, Gauwin, Reynier, Servais, 13]
Algorithmic questions

Theorem: Evaluation of FO + bounded-TC with complexity $O(|\text{formula}| \times |\text{input}| \times \#\text{variables})$

• In functional setting: it is decidable if a 2-way transducer is recognisable by a 1-way transducer [Filiot, Gauwin, Reynier, Servais, 13]

• What about the general setting, with pebbles?
Algorithmic questions

Theorem: Evaluation of FO + bounded-TC with complexity $O(|\text{formula}| \times |\text{input}| \times \#\text{variables})$

• In functional setting: it is decidable if a 2-way transducer is recognisable by a 1-way transducer [Filiot, Gauwin, Reynier, Servais, 13]

• What about the general setting, with pebbles?

• Determinisability of functional non-det transducers is decidable in PTIME [Schützenberger, 75]
Algorithmic questions

Theorem: Evaluation of FO + bounded-TC with complexity $O(|\text{formula}| \times |\text{input}| \times \#\text{variables})$

- In functional setting: it is decidable if a 2-way transducer is recognisable by a 1-way transducer [Filiot, Gauwin, Reynier, Servais, 13]

- What about the general setting, with pebbles?

- Determinisability of functional non-det transducers is decidable in PTIME [Schützenberger, 75]

- Extension to the general setting?
Algorithmic questions

Theorem: Evaluation of FO + bounded-TC with complexity $O(|\text{formula}| \times |\text{input}| \times \#\text{variables})$

• In functional setting: it is decidable if a 2-way transducer is recognisable by a 1-way transducer [Fliot, Gauwin, Reynier, Servais, 13]

• What about the general setting, with pebbles?

• Determinisability of functional non-det transducers is decidable in PTIME [Schützenberger, 75]

• Extension to the general setting?

Thank you!