Logical Characterization of Weighted Pebble Walking Automata

Benjamin Monmege
Université libre de Bruxelles, Belgium

Benedikt Bollig and Paul Gastin (LSV, ENS Cachan, France)
Marc Zeitoun (LaBRI, Bordeaux University, France)

CSL-LICS 2014

Vienna - July 15, 2014
Equivalence between automata and logic

- Well-known and studied model of computation: NFA over words
- Existing extensions over trees, grids, graphs...
- Robustness of automata intrinsically linked to logical characterization
Equivalence between automata and logic

- Well-known and studied model of computation: NFA over words
- Existing extensions over trees, grids, graphs...
- Robustness of automata intrinsically linked to logical characterization

- Büchi-Elgot-Trakhtenbrot: NFA vs MSO
- Engelfriet-Hoogeboom: pebble walking automata vs FOposTC
- Droste-Gastin: weighted automata vs restricted weighted MSO
Equivalence between automata and logic

- Well-known and studied model of computation: NFA over words
- Existing extensions over trees, grids, graphs...
- Robustness of automata intrinsically linked to logical characterization

- Büchi-Elgot-Trakhtenbrot: NFA vs MSO
- Engelfriet-Hoogeboom: pebble walking automata vs FOposTC
- Droste-Gastin: weighted automata vs restricted weighted MSO

- Aim: extend Engelfriet-Hoogeboom result to the quantitative setting, relating weighted pebble walking automata with weighted FOposTC
Graphs as a general model

**Words:** $D = \{ \rightarrow \}$

computations of sequential programs

\[
\begin{align*}
a &\rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow b \rightarrow b
\end{align*}
\]
Graphs as a general model

**Words:** $D = \{ \rightarrow \}$
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:** $D = \{ \rightarrow, \bowtie \}$
computations of recursive programs, XML documents

\[ a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow b \rightarrow b \]
Graphs as a general model

Words: $D = \{\rightarrow\}$
computations of sequential programs

\[\begin{align*}
a &\rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow b \rightarrow b
\end{align*}\]

Nested words: $D = \{\rightarrow, \bowtie\}$
computations of recursive programs, XML documents

\[\begin{align*}
a &\rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow a \rightarrow b \rightarrow a \rightarrow b \rightarrow b
\end{align*}\]

Ranked trees: $D = \{\downarrow_1, \downarrow_2\}$
expressions, formulae, parse trees

\[\begin{align*}
\text{S} &\rightarrow \text{NP} \rightarrow \text{VP} \rightarrow \text{V} \rightarrow \text{N} \\
\text{S} &\rightarrow \text{NP} \rightarrow \text{VP} \rightarrow \text{V} \rightarrow \text{N} \\
\text{John} &\rightarrow \text{hit} \rightarrow \text{the} \rightarrow \text{ball}
\end{align*}\]
Graphs as a general model

**Definition: directed graphs**

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

- \( V \) is a nonempty and finite set of vertices;
- for all edge label \( d \in D, E_d \subseteq V \times V \) is an *irreflexive relation*, describing the \( d \)-edges of the graph, which is *deterministic and codeterministic*;
- \( \lambda: V \to A \) is a vertex-labeling function;
- \( \iota \in V \) is an initial vertex.

For all edge label \( d \), we consider its reverse \( d^{-1} \) letting \( E_{d^{-1}} = (E_d)^{-1} \).
Graphs as a general model

Definition: directed graphs

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

- \( V \) is a nonempty and finite set of vertices;
- for all edge label \( d \in D \), \( E_d \subseteq V \times V \) is an irreflexive relation, describing the \( d \)-edges of the graph, which is deterministic and codeterministic;
- \( \lambda : V \to A \) is a vertex-labeling function;
- \( \iota \in V \) is an initial vertex.

For all edge label \( d \), we consider its reverse \( d^{-1} \) letting \( E_{d^{-1}} = (E_d)^{-1} \)

**Grids:** \( D = \{ \rightarrow, \uparrow \} \)

**pictures**
**Weighted pebble walking automata**

**Definition:**
- Finite number of states, initial and final states
- Ability to navigate in the graph (using the deterministic edge labels)
- Bounded supply of pebbles able to mark temporarily a position
- Pebbles are treated with a stack policy: first pebble to lift is the last dropped pebble
- Transitions equipped with weights in a complete semiring \((S, +, \times, 0, 1)\)

Examples of complete semirings:
- \((\{0, 1\}, \lor, \land, 0, 1)\)
- \((\mathbb{R}^+ \cup \{+\infty\}, +, \times, 0, 1)\)
- \((\mathbb{Z} \cup \{+\infty, -\infty\}, \min, +, +\infty, 0)\)
- \((\mathbb{Z} \cup \{+\infty, -\infty\}, \max, +, -\infty, 0)\)
- \(([0, 1], \min, \max, 1, 0)\)
- \((2^{\Sigma^*}, \cup, \cdot, \emptyset, \{\varepsilon\})\)
Weighted pebble walking automata

Definition: Semantics over \((S, +, \times, 0, 1)\)

- Configurations over a graph \(G\): \((G, q, \pi, v)\) with state \(q\), stack \(\pi\) of pebble positions and current vertex \(v\)
- Weight of a run: multiplication of the weights of transitions
- Semantics \(\llbracket A \rrbracket(G)\): sum of weights of accepting runs over \(G\)
Example of weighted pebble walking automaton

computes the biggest size of frames (empty black square)

\((\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)\)
Example of weighted pebble walking automaton

```
computes the biggest size of frames
(\mathbb{N} \cup \{-\infty\}, \text{max}, +, -\infty, 0)
```
Example of weighted pebble walking automaton

computes the biggest size of frames (empty black square)

\((\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)\)
Example of weighted pebble walking automaton

computes the biggest size of frames (empty black square)

$\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0$
Logical characterization

Classical weighted automata are **one-way** (sometimes branching) and **without pebbles**

Logical characterization for them in terms of a restricted weighted MSO logic:

- over words [Droste and Gastin, 2009]
- over trees [Droste and Vogler, 2006]
- over grids [Fichtner, 2011]
- over nested words [Mathissen, 2010]...
Logical characterization

Classical weighted automata are **one-way** (sometimes branching) and **without pebbles**

Logical characterization for them in terms of a restricted weighted MSO logic:

- over words [Droste and Gastin, 2009]
- over trees [Droste and Vogler, 2006]
- over grids [Fichtner, 2011]
- over nested words [Mathissen, 2010]...

Restricted weighted MSO does not even contain full weighted FO a priori

**Theorem: Our contribution**

Weighted pebble walking automata over graphs (wPWA) = wFOTC
Weighted first-order logic

Definition:

Classical first-order logic

\[ \varphi ::= \top \mid (x = y) \mid \text{init}(x) \mid P_a(x) \mid R_d(x, y) \mid R_d^+(x, y) \mid \neg \varphi \mid \varphi \lor \varphi \mid \exists x \varphi \]
Weighted first-order logic

**Definition:**

Classical first-order logic

\[ \varphi ::= \top | (x = y) | \text{init}(x) | P_a(x) | R_d(x, y) | R_d^+(x, y) | \neg \varphi | \varphi \vee \varphi | \exists x \varphi \]

Weighted first-order logic over graphs with weights in a semiring \((S, +, \times, 0, 1)\)

\[ \Phi ::= s | \varphi ? \Phi : \Phi | \Phi \oplus \Phi | \Phi \otimes \Phi | \bigoplus_x \Phi | \bigotimes_x \Phi \]
Weighted first-order logic

Definition:

Classical first-order logic

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

Weighted first-order logic over graphs with weights in a semiring \((S, +, \times, 0, 1)\)

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

Semantics over a graph \(G\) and a valuation \(\sigma\) of free variables

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

\[ \llbracket \Phi_1 \oplus \Phi_2 \rrbracket (G, \sigma) = \llbracket \Phi_1 \rrbracket (G, \sigma) + \llbracket \Phi_2 \rrbracket (G, \sigma) \quad \llbracket \bigoplus_x \Phi \rrbracket (G, \sigma) = \sum_{v \in V} \llbracket \Phi \rrbracket (G, \sigma[x \mapsto v]) \]

\[ \llbracket \Phi_1 \otimes \Phi_2 \rrbracket (G, \sigma) = \llbracket \Phi_1 \rrbracket (G, \sigma) \times \llbracket \Phi_2 \rrbracket (G, \sigma) \quad \llbracket \bigotimes_x \Phi \rrbracket (G, \sigma) = \prod_{v \in V} \llbracket \Phi \rrbracket (G, \sigma[x \mapsto v]) \]
Weighted first-order logic

**Definition:**

Classical first-order logic

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

Weighted first-order logic over graphs with weights in a semiring \((S, +, \times, 0, 1)\)

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

Semantics over a graph \(G\) and a valuation \(\sigma\) of free variables

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

\[
\llbracket \Phi_1 \oplus \Phi_2 \rrbracket(G, \sigma) = \llbracket \Phi_1 \rrbracket(G, \sigma) + \llbracket \Phi_2 \rrbracket(G, \sigma) \\
\llbracket \bigoplus_x \Phi \rrbracket(G, \sigma) = \sum_{v \in V} \llbracket \Phi \rrbracket(G, \sigma[x \mapsto v])
\]

\[
\llbracket \Phi_1 \otimes \Phi_2 \rrbracket(G, \sigma) = \llbracket \Phi_1 \rrbracket(G, \sigma) \times \llbracket \Phi_2 \rrbracket(G, \sigma) \\
\llbracket \bigotimes_x \Phi \rrbracket(G, \sigma) = \prod_{v \in V} \llbracket \Phi \rrbracket(G, \sigma[x \mapsto v])
\]

Examples in \((\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)\)

\[
\Phi_b = \bigotimes_x P_b(x) \? 1 : 0 \\
\Phi_w = \bigotimes_x P_w(x) \? 1 : 0 \\
\Phi_b \oplus \Phi_w
\]
Transitive closure in graphs

Binary predicate $R_\uparrow(x, y) = \exists z[R_\rightarrow(x, z) \land R_\uparrow(z, y)]$
Transitive Closure $\text{TC}_{x,y}R_\uparrow(x, y)$
  test if square (not doable in FO)
Transitive closure in graphs

Binary predicate $R^\triangleright(x, y) = \exists z[R_\rightarrow(x, z) \land R_\uparrow(z, y)]$

Transitive Closure $TC_{x,y}R^\triangleright(x, y)$

test if square (not doable in FO)

\[ TC_{x,y}[R^\triangleright(x, y) \land 1 : -\infty] \]

verifies that it is a square and computes the length of its diagonal

Weighted transitive closure: semiring $(\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$

\[
TC_{x,y}[R^\triangleright(x, y) \land 1 : -\infty]
\]

verifies that it is a square and computes the length of its diagonal
Transitive closure in graphs

Binary predicate $R_\rightarrow(x, y) = \exists z[R_\rightarrow(x, z) \wedge R_\rightarrow(z, y)]$

Transitive Closure $\text{TC}_{x, y} R_\rightarrow(x, y)$
test if square (not doable in FO)

Weighted transitive closure: semiring $(\mathbb{N} \cup \{-\infty\}, \text{max}, +, -\infty, 0)$

$$\text{TC}_{x, y}[R_\rightarrow(x, y) ? 1 : -\infty]$$

verifies that it is a square and computes the length of its diagonal

Semantics given in a complete semiring $(S, +, \times, 0, 1)$

$$[[\text{TC}_{x, y} \Phi](x', y')](G, \sigma) = \sum \prod_{\nu_0, \nu_1, \ldots, \nu_m (m>0) \ 0 \leq k \leq m-1 \ \sigma(x')=\nu_0, \sigma(y')=\nu_m} [[\Phi]](G, \sigma[x \mapsto \nu_k, y \mapsto \nu_{k+1}])$$

sum along sequences of stop-vertices

multiplication along the sequence
Bounding the Transitive Closure

- A necessary restriction to obtain a fragment of logic expressively equivalent to \( wPWA \)
- But not so restrictive in most of the cases!

\[
TC^N_{x,y} \Phi(x, y) = TC_{x,y}[\text{dist}(x, y) \leq N \ ? \ \Phi(x, y) : 0]
\]
Bounding the Transitive Closure

- A necessary restriction to obtain a fragment of logic expressively equivalent to wPWA
- But not so restrictive in most of the cases!

\[ \text{TC}_{x,y}^N \Phi(x, y) = \text{TC}_{x,y}[\text{dist}(x, y) \leq N \ ? \Phi(x, y) : 0] \]

Previous example: \( \text{TC}_{x,y}[R^\uparrow(x, y) \ ? \ 1 : -\infty] = \text{TC}_{x,y}^2[R^\uparrow(x, y) \ ? \ 1 : -\infty] \)
Bounding the Transitive Closure

- A necessary restriction to obtain a fragment of logic expressively equivalent to \text{wPWA}
- But not so restrictive in most of the cases!

\[ \text{TC}^N_{x,y} \Phi(x, y) = \text{TC}_{x,y} \left[ \text{dist}(x, y) \leq N \right. \left. ? \Phi(x, y) : 0 \right] \]

Previous example: \( \text{TC}_{x,y} \left[ R^\uparrow(x, y) ? 1 : -\infty \right] = \text{TC}^2_{x,y} \left[ R^\uparrow(x, y) ? 1 : -\infty \right] \)

**Definition: Logic \text{wFOTC}**

\[ \Phi ::= s \mid \varphi \ ? \Phi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus_x \Phi \mid \bigotimes_x \Phi \mid \text{TC}^N_{x,y} \Phi \]

with \( s \in S, \varphi \in \text{FO}, x, y \in \text{Var} \) and \( N \in \mathbb{N} \setminus \{0\} \).

Comparison with restricted \text{wMSO}:

- unrestricted use of \( \bigoplus_x \) and \( \bigotimes \), presence of \( \text{TC}_{x,y} \), absence of \( \bigoplus_X \)
Contribution

Theorem:

*over searchable graphs:* $\text{wFOTC} \rightarrow$ weighted pebble walking automata

*over zonable graphs:* weighted pebble walking automata $\rightarrow \text{wFOTC}$
Theorem:

over searchable graphs: \( \text{wFOTC} \rightarrow \text{weighted pebble walking automata} \)

over zonable graphs: \( \text{weighted pebble walking automata} \rightarrow \text{wFOTC} \)

\( \implies \) (un)decidability and complexity results over automata transfer to \( \text{wFOTC} \)
Definition: Hypothesis: searche\textit{ble} graphs

- linear order $\leq$ on vertices with $\iota$ (initial vertex) as minimal element
- existence of a guide: walking automaton $A_G$ computing $\leq$

\textit{All previously classes of graphs are searchable}
Translation of \( \text{wFOTC in wPWA} \)

**Definition: Hypothesis:** searchable graphs

- linear order \( \leq \) on vertices with \( \iota \) (initial vertex) as minimal element
- existence of a guide: walking automaton \( A_G \) computing \( \leq \)

*All previously classes of graphs are searchable*

**Inductive translation:**

\[ \Phi \oplus \Psi \] disjoint union of automata

\[ \Phi \otimes \Psi \] reset to \( \iota \)

\[ A_\Phi \rightarrow \text{reset to } \iota \rightarrow A_\Psi \]
Translation of \( wFOTC \) in \( wPWA \)

**Definition: Hypothesis:** searchable graphs

- linear order \( \leq \) on vertices with \( \iota \) (initial vertex) as minimal element
- existence of a guide: walking automaton \( A_G \) computing \( \leq \)

All previously classes of graphs are searchable

\[ \bigoplus_x \Phi \]
Definition: Hypothesis: **searchable** graphs

- linear order $\leq$ on vertices with $i$ (initial vertex) as minimal element
- existence of a guide: walking automaton $A_G$ computing $\leq$

*All previously classes of graphs are searchable*
Translation of wFOTC in wPWA

Definition: Hypothesis: **searchable** graphs

- linear order $\leq$ on vertices with $\iota$ (initial vertex) as minimal element
- existence of a guide: walking automaton $A_\mathcal{G}$ computing $\leq$

*All previously classes of graphs are searchable*

Boolean fragment: **linear size automata** (pebble and navigation)
Translation of \textit{wFOTC} in \textit{wPWA}

**Definition:** Hypothesis: \textbf{searchable} graphs

- linear order $\leq$ on vertices with $\iota$ (initial vertex) as minimal element
- existence of a guide: walking automaton $A_G$ computing $\leq$

\textit{All previously classes of graphs are searchable}

Boolean fragment: \textbf{linear size automata} (pebble and navigation)

- disjunction $\xi = \phi \lor \psi$
- existential quantification $\xi = \exists x \, \phi$
Translation of \( w\text{FOTC} \) in \( w\text{PWA} \)

Case of a formula \( [\text{TC}_{x,y}^N \Phi(x, y)](x', y') \) with \( A \) a \( w\text{PWA} \) for \( \Phi \): construction of fresh free variables

a \( w\text{PWA} \) \( A' \) with two more layers of pebbles that does the following

1. search free variable \( x' \), and drop pebble \( x \)
2. guess a sequence \( \pi \) of moves of length \( \leq N \), follow it, and drop pebble \( y \) (then flush the sequence to save memory)
3. reset to \( \iota \) and simulate \( A \)
4. search pebble \( y \)
5. guess sequence \( \pi \) of moves of length \( \leq N \), follow it, check that it holds \( x \)
   ▶ test that \( \pi \) is minimal amongst all sequences going from \( y \) to \( x \)
6. lift pebbles \( y \) and \( x \) (hence returning to the vertex of \( x \))
7. follow \( \pi \) to reach back the vertex that held \( y \), and drop pebble \( x \)
8. if \( y' \) is held by the current vertex, enter a final state
9. in every case, go back to step 2
Translation of \( \text{wFOTC in wPWA} \)

Case of a formula \( [TC_{x,y}^N \Phi(x, y)](x', y') \) with \( \mathcal{A} \) a wPWA for \( \Phi \): construction of fresh free variables

a wPWA \( \mathcal{A}' \) with two more layers of pebbles that does the following

1. search free variable \( x' \), and drop pebble \( x \)
2. guess a sequence \( \pi \) of moves of length \( \leq N \), follow it, and drop pebble \( y \)  
   \( \text{(then flush the sequence to save memory)} \)

3. reset to \( \iota \) and simulate \( \mathcal{A} \)
4. search pebble \( y \)
5. guess sequence \( \pi \) of moves of length \( \leq N \), follow it, check that it holds \( x \)

6. lift pebbles \( y \) and \( x \) (hence returning to the vertex of \( x \))
7. follow \( \pi^R \) to reach back the vertex that held \( y \), and drop pebble \( x \)
8. if \( y' \) is held by the current vertex, enter a final state
9. in every case, go back to step 2
Translation of \( wFOTC \) in \( wPWA \)

Case of a formula \( [TC_{x,y}^N \Phi(x, y)](x', y') \) with \( \mathcal{A} \) a \( wPWA \) for \( \Phi \): construction of a \( wPWA \) \( \mathcal{A}' \) with two more layers of pebbles that does the following

1. search free variable \( x' \), and drop pebble \( x \)
2. guess a sequence \( \pi \) of moves of length \( \leq N \), follow it, and drop pebble \( y \) 
   (then flush the sequence to save memory)
   ▶ test that \( \pi \) is minimal amongst all sequences going from \( x \) to \( y \)
3. reset to \( \iota \) and simulate \( \mathcal{A} \)
4. search pebble \( y \)
5. guess sequence \( \pi \) of moves of length \( \leq N \), follow it, check that it holds \( x \)
   ▶ test that \( \pi \) is minimal amongst all sequences going \( q \) from \( y \) to \( x \)
6. lift pebbles \( y \) and \( x \) (hence returning to the vertex of \( x \))
7. follow \( \pi^R \) to reach back the vertex that held \( y \), and drop pebble \( x \)
8. if \( y' \) is held by the current vertex, enter a final state
9. in every case, go back to step 2
Translation of wPWA in wFOTC

Theorem:
Let $\mathcal{G}$ be a zonable class of graphs. Then, for every wPWA $A$, we can construct a formula $\Phi$ of wFOTC such that for every graph $G \in \mathcal{G}$, and valuation $\sigma$ of free variables, $[A](G, \sigma) = [\Phi](G, \sigma)$.

Translation depends on the class $\mathcal{G}$

for a zonable class of graphs $\mathcal{G}$
Translation of \textit{wPWA in wFOTC}

**Theorem:**
Let $\mathcal{G}$ be a \textbf{zonable} class of graphs. Then, for every \textit{wPWA} $\mathcal{A}$, we can construct a formula $\Phi$ of \textit{wFOTC} such that for every graph $G \in \mathcal{G}$, and valuation $\sigma$ of free variables, $\sem{\mathcal{A}}(G, \sigma) = \sem{\Phi}(G, \sigma)$.

**Proof in two steps:**
- For the considered class of graphs, prove the \textbf{zonability};
- \textbf{Generic} translation of automata into formulae for zonable class of graphs
Zonable classes of graphs

A zoning of a graph $G$ with parameter $N$:

- an equivalence relation $\sim$, decomposing a graph into zones of diameter bounded by a constant $M$;
- set $\mathcal{W}$ of wires = (directed) edges relating different zones;
- an injective encoding function $\text{enc}: \mathcal{W} \times \{0, \ldots, N - 1\} \rightarrow V$.
Zonable classes of graphs

A zoning of a graph $G$ with parameter $N$:

- an equivalence relation $\sim$, decomposing a graph into *zones* of diameter bounded by a constant $M$;
- set $\mathcal{W}$ of wires = (directed) edges relating different zones;
- an injective encoding function $enc: \mathcal{W} \times \{0, \ldots, N - 1\} \rightarrow V$

\begin{figure}
\centering
\includegraphics[width=\textwidth]{zone_partitioning.png}
\caption{Zone partitioning of a graph: zones are related by wires depicted with dashed convention, and the red area linked to a vertex $v$ of a zone.}
\end{figure}

**and** $\sim$ and $enc$ must be expressible by some formulae $\text{zone}(z, z')$ and $\text{enc}_n(z, z', x)$ (for $n \in \{0, \ldots, N - 1\}$) in wFOTC
Examples

\[\text{Figure 6.7: Zone partitioning of a word and description of the encoding function}\]

\[\text{Figure 6.8: Zone partitioning of a picture}\]

\[N - 1 \leq |w| - 2KN < 3N - 1\]
Examples

\[ N - 1 \leq |w| - 2kN < 3N - 1 \]
Before stating the translation theorems, we give a non-exhaustive list of zonable classes. Each reserved for the wires of one border of the zone. This defines an injection as wires are separated by a distance of \( \frac{w_{\text{FO}} + w_{\text{TC}}}{z} \) (see Figure 6.7), except the last zone that may contain at most positions: hence each zone has a diameter bounded by \( \frac{2(k-1)N}{z} \) (except the largest zone possible is the one on the right bottom corner which can have width and height bounded above by \( \frac{2kN - 1}{z} \)). Forgetting, bounded above by \( \frac{2(k+1)N}{z} \).

For every integer \( k \), they can be described using modulo computations: henceforth, we define \( \mu(z) \), \( \lambda(z) \), and \( \rho(z) \). They are computable but also trees, nested words, Mazurkiewicz traces, rings...
Translation in a zonable class of graphs

- External (bounded) transitive closure jumping from zone to zone: state at the wires encoded using $\text{enc}$;
- Internal (bounded) transitive closures to compute the weights of the infinite set of runs restricted to a zone: computation by McNaughton-Yamada algorithm, state directly encoded in the formulae.
Translation in a zonable class of graphs

Weight of the runs from \( z_i \) in state \( q_i \) to \( z_f \) in state \( q_f \):

\[
\bigoplus_{x',y'} \bigg[ \bigoplus_{z_1,z'_1} \bigoplus_{q_1} \text{enc}_{q_1}(z_1, z'_1, x') \otimes \Phi_{q_i,q_1}(z_i, z_1) \bigg] \otimes \big[ \text{TC}^{3M}_{y_1,y_2} \Psi \big](x', y') \\
\otimes \bigoplus_{z_2,z'_2} \bigoplus_{q_2,q'_2} \big[ \text{enc}_{q_2}(z_2, z'_2, y') \otimes \text{tr}_{q_2,q'_2}(z_2, z'_2) \otimes \Phi_{q'_2,q_f}(z'_2, z_f) \big]
\]

with \( \Psi(y_1, y_2) \) the formula

\[
\bigoplus_{z_1,z'_1} \bigoplus_{q_1,q'_1} \bigoplus_{z_2,z'_2} \bigoplus_{q_2} \big[ \text{enc}_{q_1}(z_1, z'_1, y_1) \otimes \text{tr}_{q_1,q'_1}(z_1, z'_1) \otimes \text{enc}_{q_2}(z_2, z'_2, y_2) \otimes \Phi_{q'_1,q_2}(z'_1, z_2) \big]
\]
Translation in a zonable class of graphs

Weight of the runs from $z_i$ in state $q_i$ to $z_f$ in state $q_f$:

$$
\bigoplus_{x',y'} \left[ \bigoplus_{z_1,z_1', q_1 \in Q} \bigoplus \text{enc}_{q_1}(z_1, z_1', x') \otimes \Phi_{q_i,q_1}(z_i, z_1) \right] \otimes \left[ \text{TC}_{y_1,y_2}^3 \Psi(x', y') \right] \\
\otimes \bigoplus_{z_2,z_2', q_2,q_2' \in Q} \left[ \text{enc}_{q_2}(z_2, z_2', y') \otimes \text{tr}_{q_2,q_2'}(z_2, z_2') \otimes \Phi_{q_2',q_f}(z_2', z_f) \right]
$$

with $\Psi(y_1, y_2)$ the formula

$$
\bigoplus_{z_1,z_1', q_1,q_1', q_1 \in Q} \bigoplus_{z_2,z_2', q_2 \in Q} \left[ \text{enc}_{q_1}(z_1, z_1', y_1) \otimes \text{tr}_{q_1,q_1'}(z_1, z_1') \otimes \text{enc}_{q_2}(z_2, z_2', y_2) \otimes \Phi_{q_1',q_2}(z_1', z_2) \right]
$$

$\Phi_{q,q'}(x, x')$ formula computing the weight of the runs from $x$ in $q$ to $x'$ in $q'$, staying in the zone containing both $x$ and $x'$

- built by McNaughton-Yamada algorithm, with cascade of **bounded** transitive closures (since zones have bounded diameter)
Conclusion and Perspectives

- Expressive equivalence between weighted pebble walking automata and weighted first-order logic with bounded transitive closure, over arbitrary complete semirings
- Additional reasonable requirements on the classes of graphs (searchable and zonable), met by usual examples of graphs (words, nested words, trees, grids, Mazurkiewicz traces, rings...)
- Interesting special case: a logic for graph-to-word transducers (non-commutative semiring of languages over an alphabet \( \Sigma \))

Translation from automata to logic with less transitive closures? as in [Bollig, Gastin, Monmege, and Zeitoun, 2010] for words and the non-looping semantics

Case of strong pebbles to deal with unbounded transitive closure?

Extension to infinite structures?

Thank you!
Conclusion and Perspectives

- Expressive equivalence between weighted pebble walking automata and weighted first-order logic with bounded transitive closure, over arbitrary complete semirings
- Additional reasonable requirements on the classes of graphs (searchable and zonable), met by usual examples of graphs (words, nested words, trees, grids, Mazurkiewicz traces, rings...)
- Interesting special case: a logic for graph-to-word transducers (non-commutative semiring of languages over an alphabet $\Sigma$)
- Translation from automata to logic with less transitive closures? as in [Bollig, Gastin, Monmege, and Zeitoun, 2010] for words and the non-looping semantics
- Case of strong pebbles to deal with unbounded transitive closure?
- Extension to infinite structures?
Conclusion and Perspectives

- Expressive equivalence between weighted pebble walking automata and weighted first-order logic with bounded transitive closure, over arbitrary complete semirings
- Additional reasonable requirements on the classes of graphs (searchable and zonable), met by usual examples of graphs (words, nested words, trees, grids, Mazurkiewicz traces, rings...)
- Interesting special case: a logic for graph-to-word transducers (non-commutative semiring of languages over an alphabet $\Sigma$)

- Translation from automata to logic with less transitive closures? as in [Bollig, Gastin, Monmege, and Zeitoun, 2010] for words and the non-looping semantics
- Case of strong pebbles to deal with unbounded transitive closure?
- Extension to infinite structures?

Thank you!


