Logical Characterization of Weighted Pebble Automata Navigating over Graphs

Benjamin Monmege
Université Libre de Bruxelles, Belgium

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

7th International Workshop Workshop WATA 2014

Leipzig - May 5, 2014

To be presented at CSL-LICS 2014
Weighted Pebble Walking Automata

- Unusual mechanism
- Expressive power not fully clear

**AIM:** study expressive power in terms of other formalisms, e.g., of logic
Weighted Pebble Walking Automata

- Unusual mechanism
- Expressive power not fully clear

**AIM:** study expressive power in terms of other formalisms, e.g., of logic

Many such results for weighted automata: over words [Droste and Gastin, 2009], over trees [Droste and Vogler, 2006], over grids [Fichtner, 2011], over nested words [Mathissen, 2010]...
Weighted Pebble Walking Automata

- Unusual mechanism
- Expressive power not fully clear

**AIM:** study expressive power in terms of other formalisms, e.g., of logic

Many such results for weighted automata: over words [Droste and Gastin, 2009], over trees [Droste and Vogler, 2006], over grids [Fichtner, 2011], over nested words [Mathissen, 2010]...

**Boolean setting** [Engelfriet and Hoogeboom, 2007]

Pebble Walking Automata = FO + posTC
Weighted Pebble Walking Automata

- Unusual mechanism
- Expressive power not fully clear

**AIM:** study expressive power in terms of other formalisms, e.g., of **logic**

Many such results for weighted automata: over words [Droste and Gastin, 2009], over trees [Droste and Vogler, 2006], over grids [Fichtner, 2011], over nested words [Mathissen, 2010]...

**Boolean setting [Engelfriet and Hoogeboom, 2007]**

Pebble Walking Automata = FO + posTC

Extension in the quantitative setting

**Theorem:**

Weighted Pebble Walking Automata (wPWA) = wFOTC
Binary predicate $R^\uparrow(x, y) = \exists z[R_{\to}(x, z) \land R^\uparrow(z, y)]$

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

test if *square* (not doable in FO)
Binary predicate \( R_\uparrow(x, y) = \exists z[R_\rightarrow(x, z) \land R_\uparrow(z, y)] \)

Transitive Closure \( TC_{x,y}R_\uparrow(x, y) \)

test if square (not doable in FO)

Weighted Transitive Closure: semiring \((\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)\)

\[
TC_{x,y}[R_\uparrow(x, y) ? 1 : -\infty]
\]

verifies that it is a square and computes the length of its diagonal
Transitive Closure in Graphs

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

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

test if square (not doable in FO)

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

$$TC_{x,y}[R^\uparrow(x, y) ? 1 : -\infty]$$

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

Semantics of Weighted Transitive Closure: complete semiring $(\mathcal{S}, +, \times, 0, 1)$

$$[[TC_{x,y}\Phi](x', y')](G, \sigma) = \sum_{v_0,v_1,\ldots,v_m \ (m>0) \ 0 \leq k \leq m-1} \prod_{\sigma(x')=v_0, \sigma(y')=v_m} [[\Phi]](G, \sigma[x \mapsto v_k, y \mapsto v_{k+1}])$$

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!

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

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

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

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

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

**Definition: Logic wFOTC**

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

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

Inductive construction for searchable graphs

- For the wFO fragment, see Paul’s talk
- Case of a formula $\text{TC}^N_{x,y} \Phi(x,y)(x',y')$ 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 of moves of length $\leq N$, follow it, and drop pebble $y$ (then flush the sequence to save memory)
  3. goes back to the initial vertex and simulate $\mathcal{A}$
  4. search pebble $y$
  5. guess a 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

fresh free variables
Translation of wFOTC in wPWA

Inductive construction for searchable graphs

- For the wFO fragment, see Paul’s talk
- Case of a formula $[\text{TC}_{x,y}^N \Phi(x,y)](x',y')$ 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 of moves of length $\leq N$, follow it, and drop pebble $y$
     (then flush the sequence to save memory)

  3. **goes back to the initial vertex** and simulate $\mathcal{A}$
  4. **search** pebble $y$
  5. guess a 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

Inductive construction for searchable graphs

- For the wFO fragment, see Paul’s talk
- Case of a formula \([\text{TC}_{x,y}^N \Phi(x,y)](x',y')\) with \(A\) a wPWA for \(\Phi\):

  construction of a wPWA \(A'\) 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. goes back to the initial vertex and simulate \(A\)
4. **search** pebble \(y\)
5. guess a 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 \(\pi^R\) 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 \textit{wPWA} in \textit{wFOTC}

\textbf{Theorem:} \par

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, $[[\mathcal{A}]](G, \sigma) = [[\Phi]](G, \sigma)$.

\begin{itemize}
  \item Proof in two steps:
  \begin{itemize}
    \item For the considered class of graphs, prove the \textit{zonability};
    \item Generic translation of automata into formulae for zonable class of graphs
  \end{itemize}
\end{itemize}

Examples of zonable classes of graphs: words, trees, grids/pictures, nested words, Mazurkiewicz traces...
**Translation of wPWA in wFOTC**

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

**Proof in two steps:**

- For the considered class of graphs, prove the **zonability**;
- **Generic** translation of automata into formulae for zonable class of graphs

Example of zonable classes of graphs: words, trees, grids/pictures, nested words, Mazurkiewicz traces...
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 $= (\text{directed})$ edges relating different zones;
- an injective encoding function $\text{enc}: \mathcal{W} \times \{0, \ldots, N - 1\} \to V$
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 $= \text{(directed)}$ edges relating different zones;
- an injective encoding function $\text{enc} : \mathcal{W} \times \{0, \ldots, N - 1\} \rightarrow V$

![Diagram of zones and wires]

and $\sim$ and $\text{enc}$ 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: words and grids
Examples: words and grids

- Formula of each zone has a height bounded by height less than
- In ranked trees, we consider zones to be subtrees of height at least
  - Trees show that for every
  - Deciding of a decodable order between the wires, it is easy to design a formula
- Bounded above by
  - Pictures will be square subpictures of width
  - Similar ideas to cut pictures into zones have been used for other purposes in [Mat98]. Zones
  - Separated by a distance of
  - Moreover, wires will simply be edges of the form
    - Formula of graphs.
    - Which is an unambiguous formula as
      - Then we can consider that there is a single zone containing all the vertices, and hence no
  - For the sake of simplicity, about the zones on the right and on the bottom, we obtain as
  - Formulas same zone (z1, z2):
    - Forgetting,
  - Each zone (except the larger ones) has at most
  - They can be described using modulo computations: henceforth, we define
  - They relate two distinct zones of the graph. Hence, they can
  - Which has
  - This defines an injection as wires are
  - Each zone has a diameter bounded
  - Then we can consider that there is a single zone containing all the vertices, and hence no
  - Forgetting,
  - Each zone has at most
  - It has to be noticed that wires relate two distinct zones of the graph. Hence, they can
  - Each zone has a height bounded
  - In ranked trees, we consider zones to be subtrees of height at least
  - Trees show that for every
  - Deciding of a decodable order between the wires, it is easy to design a formula
  - A -zonable class of graphs.
  - Formulas same zone (z1, z2):
    - Forgetting,
  - Each zone (except the larger ones) has at most
  - They can be described using modulo computations: henceforth, we define
  - They relate two distinct zones of the graph. Hence, they can
  - Which has
  - This defines an injection as wires are
  - Each zone has a diameter bounded
  - Then we can consider that there is a single zone containing all the vertices, and hence no
  - Forgetting,
  - Each zone has a height bounded
  - In ranked trees, we consider zones to be subtrees of height at least
  - Trees show that for every
  - Deciding of a decodable order between the wires, it is easy to design a formula
  - A -zonable class of graphs.
  - Formulas same zone (z1, z2):
    - Forgetting,
  - Each zone (except the larger ones) has at most
  - They can be described using modulo computations: henceforth, we define
  - They relate two distinct zones of the graph. Hence, they can
  - Which has
  - This defines an injection as wires are
  - Each zone has a diameter bounded
  - Then we can consider that there is a single zone containing all the vertices, and hence no
  - Forgetting,
  - Each zone has a height bounded
  - In ranked trees, we consider zones to be subtrees of height at least
  - Trees show that for every
  - Deciding of a decodable order between the wires, it is easy to design a formula
  - A -zonable class of graphs.
  - Formulas same zone (z1, z2):
    - Forgetting,
  - Each zone (except the larger ones) has at most
  - They can be described using modulo computations: henceforth, we define
  - They relate two distinct zones of the graph. Hence, they can
  - Which has
  - This defines an injection as wires are
  - Each zone has a diameter bounded
  - Then we can consider that there is a single zone containing all the vertices, and hence no
  - Forgetting,
  - Each zone has a height bounded
  - In ranked trees, we consider zones to be subtrees of height at least
  - Trees show that for every
  - Deciding of a decodable order between the wires, it is easy to design a formula
  - A -zonable class of graphs.
  - Formulas same zone (z1, z2):
    - Forgetting,
  - Each zone (except the larger ones) has at most
  - They can be described using modulo computations: henceforth, we define
  - They relate two distinct zones of the graph. Hence, they can
  - Which has
  - This defines an injection as wires are
  - Each zone has a diameter bounded
  - Then we can consider that there is a single zone containing all the vertices, and hence no
  - Forgetting,
Translation in a zonable class of graphs

- External (bounded) transitive closure jumping from zone to zone: state at the wires encoded using $enc$;
- 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 $q_f$:

$$\bigoplus_{x',y'} \bigoplus_{z_1,z_1',q_1 \in Q} \text{enc}_{q_1}(z_1, z_1', x') \otimes \Phi_{q_i,q_1}(z_i, z_1) \otimes [\text{TC}^M_{y_1,y_2} \Psi](x', y')$$

$$\otimes \bigoplus_{z_2,z_2',q_2,q_2' \in Q} \left[ \text{enc}_{q_2}(z_2, z_2', y') \otimes \text{tr}_{q_2,q_2'}(z_2, z_2') \otimes \Phi_{q_2,q_f}(z_2', z_f) \right]$$

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

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

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

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

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

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

$\Phi_{q,q'}(x,x')$ 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 continuous 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...).
- Interesting special case: graph-to-word transducers (non-commutative semiring of languages over an alphabet Σ).

- 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?
References


