Pebble transducers with unary output

Gaëtan Douéneau-Tabot (IRIF, Université de Paris)
doueneau@irif.fr
Paris, June 29th, 2021
Motivation: transducers are simple programs

Finite automata
→ Model simple programs that accept/reject: $A^* \rightarrow \{\top, \bot\}$
  ▶ All models (two-way, one-way, etc.) are equivalent
  ▶ "Everything" is decidable: equivalence, membership, etc.

Deterministic transducers = finite automata + outputs
→ Model simple programs that produce an output: $A^* \rightarrow B^*$
  ▶ Different models have different expressive power
  ▶ Many questions are more complex
Contributions: membership results

Class membership problems

► Decide if a function described by a transducer from a class $\mathcal{C}$ can be computed by a transducer in some "simpler" class $\mathcal{C}'$.

► It is program optimization!

We study membership problems for pebble transducers
1. Regular and polyregular functions

2. Two variants of pebble transducers

3. Pebble transducers with unary output
Regular and polyregular functions
Two-way transducer = two-way automaton + outputs

→ Computes a function $f : A^* \rightarrow B^*$
Regular functions = functions computed by two-way transducers

Example: regular functions

- remove-a : abaacdaaa → bcd
- reverse : abcd → dcba (right-to-left reading, detect the borders)

Some properties of regular functions

- closure under composition [Chytil and Jákl, 1977]
- logical specifications (MSO-transductions) [Engelfriet and Hoogeboom, 2001]
- equivalent to copyless streaming string transducers [Alur and Cerný, 2010]
- decidable equivalence problem for two-way transducers [Gurari, 1982]
Deterministic pebble transducers

Ingredients: nested two-way transducers + pebbles

Definition: 0-pebble transducer = two-way transducer

Definition: 1-pebble transducer

→ $k$-pebble: stack of submachines of depth $k$
Polyregular functions = functions computed by \( k \)-pebble transducers (for some \( k \geq 0 \)). With \( k = 0 \) it contains regular functions.

**Example: functions computable by 1-pebble**

- **u-square**: \( a^n \mapsto a^{n^2} \)
- **#-square**: \( aaaa \mapsto aaaa###aaaa###aaaa# \)
- **#-pref**: \( aaaa \mapsto a###aa###aaa# \)
- **square**: \( aaaa \mapsto _aaaa###aaa###aaa#aaa# \)
Polyregular functions = functions computed by $k$-pebble transducers (for some $k \geq 0$) With $k = 0$ it contains regular functions.

Example: functions computable by 1-pebble

- u-square: $a^n \rightarrow a^{n^2}$
- #-square: $aaaa \rightarrow aaaaa#aaaaa#aaaaa#aaaaa$
- #-pref: $aaaa \rightarrow a##aa##aaa##aaaa$
- square: $aaaa \rightarrow _aaaa##_aaa##_aaa##_aaa#$
Polyregular functions = functions computed by $k$-pebble transducers (for some $k \geq 0$) With $k = 0$ it contains regular functions.

**Example: functions computable by 1-pebble**

- **u-square**: $a^n \rightarrow a^{n^2}$
- **#-square**: $aaaa \rightarrow aaaa#aaaa#aaaa#aaaa#$
- **#-pref**: $aaaa \rightarrow a#aa#aaa#aaaa#$
- **square**: $aaaa \rightarrow _aaaa#_aaaa#_aaaa#_aaaa#$
Polyregular functions = functions computed by \( k \)-pebble transducers (for some \( k \geq 0 \)) With \( k = 0 \) it contains regular functions.

Example: functions computable by 1-pebble

- u-square: \( a^n \rightarrow a^{n^2} \)
- #\text{-square}: \text{aaaa} \rightarrow \text{aaaa#aaaa#aaaa#aaaa#}
- #\text{-pref}: \text{aaaa} \rightarrow a#aa#aa#aaaa#
- square: \text{aaaa} \rightarrow _{aaaa#aaaa#aaaa#aaaa#}
Polyregular functions

Polyregular functions = functions computed by $k$-pebble transducers
(for some $k \geq 0$) With $k = 0$ it contains regular functions.

Example: functions computable by 1-pebble

- $u$-square: $a^n \mapsto a^{n^2}$
- $#$-square: $aaaa \mapsto aaaa#aaaa#aaaa#aaaa#$
- $#$-pref: $aaaa \mapsto a#aa#aaa#aaaa#$
- square: $aaaa \mapsto _aaaa#_aaa#aaaaa#_aaa#$

![Diagram of 1-pebble transducer for example functions]
Main results on polyregular functions

- closure under composition [Engelfriet, 2015]
- logical specifications (MSO-interpretations) [Bojańczyk et al., 2019]
- "pebble optimisation" (computing the minimal number of pebbles) [Lhote, 2020]

¿ equivalence problem of pebble transducers ¿
Two variants of pebble transducers
Variants of pebble transducers

Reminder: 1-pebble transducer

Why have variants been studied?

- natural restrictions of pebble transducers;
- equivalence with other models
  (register transducers [Douéneau-Tabot et al., 2020], \( \lambda \)-calculus [Nguyễn et al., 2021], etc.)
Definition: 1-blind transducer

$k$-blind transducer $=$ $k$-pebble transducer without pebbles

Subclasses

✓ a $k$-blind transducer can be simulated by a $k$-pebble transducer
Definition: 1-marble transducer

→ the submachine works on a prefix of the input

Subclasses

✓ a $k$-marble transducer can be simulated by a $k$-pebble transducer
Strict inclusions for $k = 1$?

Classes of functions [Douéneau-Tabot et al., 2020] [Nguyên et al., 2021]

1-pebble

- 1-blind

- 1-marble

Example: u-square: $a^n \rightarrow a^{n^2}$

- Computable by 1-blind?
- Computable by 1-marble?
Strict inclusions for $k = 1$?

Classes of functions [Douéneau-Tabot et al., 2020] [Nguyễn et al., 2021]

1-pebble

1-blind

1-marble

Example: u-square: $a^n \rightarrow a^{n^2}$

- Computable by 1-blind?

- Computable by 1-marble?
Strict inclusions for $k = 1$?

Classes of functions [Douéneau-Tabot et al., 2020] [Nguyễn et al., 2021]

1-pebble

1-blind

1-marble

Example: \(-\text{pref} : aaaa \rightarrow a\#aa\#aaa\#aaaa\#

- Computable by 1-blind?
- Computable by 1-marble?
Strict inclusions for $k = 1$?

Classes of functions [Douéneau-Tabot et al., 2020] [Nguyên et al., 2021]

1-pebble

1-blind

1-marble

u-square: $a^n \rightarrow a^{n^2}$

#-pref: $aaaa \rightarrow a#aa#aaa#aaaa#$

Example:  

#-pref: $aaaa \rightarrow a#aa#aaa#aaaa#$

- Computable by 1-blind?  
  X not computable

- Computable by 1-marble?
Strict inclusions for $k = 1$?

Classes of functions [Douéneau-Tabot et al., 2020] [Nguyễn et al., 2021]

1-pebble

1-blind

1-marble

- $\text{u-square: } a^n \rightarrow a^{n^2}$
- $\text{#-pref: } aaaa \rightarrow a#aa#aaa#a###$ 

Example:  

$\text{#-square: } aaaa \rightarrow aaaa#aaaa#aaaa#aaaa#$

- Computable by 1-blind?
- Computable by 1-marble?
Strict inclusions for $k = 1$?

**Classes of functions [Douéneau-Tabot et al., 2020] [Nguyễn et al., 2021]**

- **1-pebble**
  - **1-blind**
  - **1-marble**

### Example:

- #-square: $aaaa \rightarrow aaaa#aaaa#aaaa#aaaa#$
- u-square: $a^n \rightarrow a^{n^2}$
- #-pref: $aaaa \rightarrow a#aa#aaa#aaaa#$

**Computable by 1-blind?**

**Computable by 1-marble?**

*X not computable*
Strict inclusions for $k = 1$?

Classes of functions [Douéneau-Tabot et al., 2020] [Nguyễn et al., 2021]

<table>
<thead>
<tr>
<th>1-pebble</th>
<th>1-blind</th>
<th>1-marble</th>
</tr>
</thead>
<tbody>
<tr>
<td>square: $aaaa \rightarrow _{aaaa#_aaaa#_aaaa#_aaaa#}$</td>
<td>#-square: $aaaa \rightarrow _{aaaa#_aaaa#_aaaa#_aaaa#}$</td>
<td>u-square: $a^n \rightarrow a^{n^2}$</td>
</tr>
<tr>
<td>#-pref: $aaaa \rightarrow _{a#aa#_aaa#_aaa#}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example: square: $aaaa \rightarrow _{aaaa#_aaaa#_aaaa#_aaaa#}$

- Computable by 1-blind? $\times$ not computable
- Computable by 1-marble? $\times$ not computable
Pebble transducers with unary output
Transducers with unary output

From now on, machines have a **unary output alphabet**
In other words $f : A^* \rightarrow \mathbb{N}$

→ Goal: study membership problems
Pebbles and marbles the same

**Theorem:** \( k\)-pebble = \( k\)-marble

\( k\)-pebble transducers and \( k\)-marble transducers with unary output compute the same functions.

(it coincides with rational series of polynomial growth)

!! the conversion may modify the "origin semantics" !!

**Corollary:**

Equivalence of pebble transducers with unary output is decidable.
Pebbles and marbles are the same

**Example:** $u$-square: $a^n \rightarrow n^2$

**Example:** $\text{prod} : w \in \{a, b\}^* \rightarrow |w|_a \times |w|_b$

\begin{itemize}
  \item 1-pebble
  \begin{align*}
    a & \quad b & \quad a & \quad a & \quad b \\
    1 & \quad 1 & \quad 1 & \quad 1 & \quad 1 \\
  \end{align*}
  \rightarrow \text{for each } b, \text{ compute the number of } a
\end{itemize}

\begin{itemize}
  \item 1-marble
  \begin{align*}
    a & \quad b & \quad a & \quad a & \quad b \\
    1 & \quad 1 & \quad 1 & \quad 1 & \quad 1 \\
  \end{align*}
\end{itemize}
Pebbles and marbles are the same

Example: \( u\text{-square} : \quad a^n \mapsto n^2 \)

Example: \( \text{prod} : \quad w \in \{a, b\}^* \mapsto |w|_a \times |w|_b \)

\[\begin{align*}
\text{1-pebble} & \quad \text{1-marble} \\
\begin{array}{ccccc}
a & b & a & a & b \\
\end{array} & \quad \begin{array}{ccccc}
a & b & a & a & b \\
\end{array} \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{align*}\]

→ for each \( b \), compute the number of \( a \)

→ for each \( b \), compute the number of \( a \) before it
→ for each \( a \), compute the number of \( b \) before it
Classes of functions with unary output

1-pebble = 1-marble

1-blind

\[
\text{prod: } w \in \{a, b\}^* \mapsto |w|_a \times |w|_b
\]

\[
\text{u-square: } a^n \mapsto n^2
\]
Strict inclusions for $k = 1$?

Classes of functions with unary output

1-pebble = 1-marble

1-blind

it-square: $a^n b \cdots b a^n b a^n b b a^n b \mapsto \sum_{i=1}^{\ell} (n_i)^2$

triangle: $a^n b \cdots b a^n b a^n b b a^n b \mapsto \sum_{i=1}^{\ell} i n_i$

prod: $w \in \{a, b\}^* \mapsto |w|_a \times |w|_b$

u-square: $a^n \mapsto n^2$

**Theorem:**
One can decide if a function with unary output given by 1-pebble transducer is computable by a 1-blind transducer.

!! the conversion may modify the "origin semantics" !!
Intuitions on the decision procedure

Technical tools

- transition monoids of two-way transducers
- monoid theory, including factorization forests

→ We give a decidable characterization of 1-marble (= 1-pebble) transducers which describe a function computable by 1-blind.
Strict inclusions for $k = 1$

Pumping-like lemma

If $f$ is computable by a $k$-blind transducer for some $k \geq 0$, then for all $\alpha, \beta \in A^*$, $u \in A^+$, there exists $\omega > 0$ such that if $w = \alpha u^\omega \beta$ then

$$X, Y \mapsto f\left(w^{2\omega-1}(\alpha u^\omega X \beta)w^{\omega-1}(\alpha u^\omega Y \beta)w^\omega\right)$$

is a polynomial in $X + Y$ for $X, Y \geq 3$. 
Strict inclusions for $k = 1$

Pumping-like lemma
If $f$ is computable by a $k$-blind transducer for some $k \geq 0$, then for all $\alpha, \beta \in A^*, u \in A^+$, there exists $\omega > 0$ such that if $w = \alpha u^\omega \beta$ then

$$X, Y \rightarrow f \left( w^{2\omega-1}(\alpha u^\omega X \beta)w^{\omega-1}(\alpha u^\omega Y \beta)w^\omega \right)$$

is a polynomial in $X + Y$ for $X, Y \geq 3$.

Application: it-square: $a^{n_1} b \cdots ba^{n_2} ba^{n_1} b \rightarrow \sum_{i=1}^{\ell} (n_i)^2$

$$\text{it-square} \left( (ba^\omega b)^{2\omega-1}(ba^\omega X b)(ba^\omega b)^{\omega-1}(ba^\omega Y b)(ba^\omega b)^\omega \right)$$

is not a polynomial in $X + Y$ for any $\omega > 0$. 
Example 1: two functions computed by 1-marble transducers

- **u-square**: $a^n \rightarrow n^2$
  
  $\begin{array}{ccccc}
  a & a & a & a & a \\
  1 & 2 & 1 & 2 & 1 \\
  2 & 2 & 2 & 2 & 2 \\
  \end{array}$

- **it-square**: $a^n b \cdots b a^n b a^{n_1} b \rightarrow \sum_{i=1}^{\ell} (n_i)^2$
  
  $\begin{array}{cccc}
  a & b & a & a \\
  1 & 2 & 1 & 1 \\
  \end{array}$
Necessary conditions on "blocks"

Example 1: two functions computed by 1-marble transducers

- **u-square**: $a^n \rightarrow n^2$
  - From $a$ to $a$: 2
  - Computable by 1-blind

- **it-square**: $a^{n_1} b \cdots b a^{n_2} b a^{n_1} b \rightarrow \sum_{i=1}^{\ell} (n_i)^2$
  - From $a$ to $a$:
    - $\rightarrow 2$ if in same block $a^{n_i}$
    - $\rightarrow 0$ otherwise
  - Not $k$-blind computable
Example 2: two functions computed by 1-marble transducers

- **prod**: \( w \in \{a, b\}^* \rightarrow |w|_a \times |w|_b 

\[
\begin{array}{cccccc}
\cdot & b & a & a & a & b \\
\hline
1 & 1 & 1 & 1 & \\
\end{array}
\]

- **triangle**: \( a^n \ell b \cdots ba \ell_2 ba \ell_1 b \rightarrow \sum_{i=1}^{\ell} i n_i 

\[
\begin{array}{cccccc}
a & b & a & a & b \\
\hline
1 & 1 & 1 & 1 \\
\end{array}
\]
Example 2: two functions computed by 1-marble transducers

\[ \text{prod : } w \in \{a, b\}^* \rightarrow |w|_a \times |w|_b \]

- From \(a\) to \(b\) : 1
- From \(b\) to \(a\) : 1

\(\checkmark\) computable by 1-blind

\[ \text{triangle: } a^n b \ldots ba^n_2 ba^n_1 b \rightarrow \sum_{i=1}^{\ell} in_i \]

- From \(a\) to \(b\) : 0
- From \(b\) to \(a\) : 1

\(\times\) not \(k\)-blind computable
Outlook
New membership techniques

Towards other membership problems:

- $k$-pebble transducer with unary output $\rightarrow k$-blind
- $k$-pebble transducer (not unary) $\rightarrow k$-marble

- contrary to the "pebble optimization" problems of [Lhote, 2020] [Douéneau-Tabot et al., 2020] [Nguyễn et al., 2021] there is no clear intrinsic property of the function for deciding membership