MIPRCV
CONSOLIDER INGENIO 2010
Multimodal Interaction in Pattern Recognition and Computer Vision

Using Multiplicity Automata to Identify Transducer Relations from Membership and Equivalence Queries

Jose Oncina
Dept. Lenguajes y Sistemas Informáticos - Universidad de Alicante

oncina@dlsi.ua.es

September 2008

- Usually a transduction is viewed as a string to string function

$$f(\text{"My red car"}) = \text{"mi coche rojo"}$$

- A particular type of transductions is the Subsequential Transductions
  - are based on a DFA
- We have algorithms to deal with this type of transductions
  - The OSTIA algorithm: from input-output pairs
  - The Vilar algorithm: from MAT
- Sometimes we have to cope with ambiguities

$$f(\text{"My red car"}) = \text{"Mi coche (rojo + colorado + encarnado)"}$$

Motivation

- Usually a transduction is viewed as a string to string function

$$f(\text{"My red car"}) = \text{"mi coche rojo"}$$

- A particular type of transductions is the Subsequential Transductions
  - are based on a DFA
- We have algorithms to deal with this type of transductions
  - The OSTIA algorithm: from input-output pairs
  - The Vilar algorithm: from MAT
- Sometimes we have to cope with ambiguities

$$f(\text{"My red car"}) = \text{"Mi coche (rojo + colorado + encarnado)"}$$

- Usually a transduction is viewed as a string to string function

$$f(\text{"My red car"}) = \text{"mi coche rojo"}$$

- A particular type of transductions is the Subsequential Transductions
  - are based on a DFA
- We have algorithms to deal with this type of transductions
  - The OSTIA algorithm: from input-output pairs
  - The Vilar algorithm: from MAT
- Sometimes we have to cope with ambiguities

$$f(\text{"My red car"}) = \text{"Mi coche (rojo + colorado + encarnado)"}$$

- Usually a transduction is viewed as a string to string function

$$f(\text{"My red car"}) = \text{"mi coche rojo"}$$

- A particular type of transductions is the Subsequential Transductions
  - are based on a DFA
- We have algorithms to deal with this type of transductions
  - The OSTIA algorithm: from input-output pairs
  - The Vilar algorithm: from MAT
- Sometimes we have to cope with ambiguities

$$f(\text{"My red car"}) = \text{"Mi coche (rojo + colorado + encarnado)"}$$

Outline

1. Multiplicity Automata
2. Exact Learning
3. A bit of Algebra
4. Examples

Outline

1. Multiplicity Automata
2. Exact Learning
3. A bit of Algebra
4. Examples

- Multiplicity automata are essentially non deterministic stochastic automata with only one initial state and no restrictions to force the normalization

Definition (Multiplicity Automata)

A Multiplicity Automaton (MA) of size r, is:

- a set of $|\Sigma|$ $r \times r$ matrices $\{\mu_\sigma : \sigma \in \Sigma\}$ with elements of the field $\mathcal{K}$
- a row-vector $\lambda = (\lambda_1, \dots, \lambda_r) \in \mathcal{K}^r$
- a column-vector $\gamma = (\gamma_1, \dots, \gamma_r)^t \in \mathcal{K}^r$

- The MA A defines a function $f_A : \Sigma^* \to \mathcal{K}$ as:

$$f_A(x_1 \dots x_n) = \lambda \mu_{x_1} \dots \mu_{x_n} \gamma$$

Let the MA A defined by:

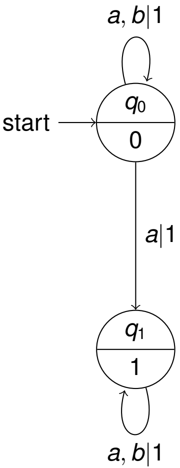
$$\lambda = \begin{pmatrix} 1 & 0 \end{pmatrix} \ \mu_a = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \ \mu_b = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \ \gamma = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

In this case:

$$\mu(x) = \mu(x_1 \dots x_n) = \mu_{x_1} \dots \mu_{x_n} = \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}$$

Where α is the number of times that a appears in x.
Then

$$f_A(x) = \lambda \mu(x) \gamma = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \alpha$$



Let $f : \Sigma^* \to \mathcal{K}$ be a function.

- The Hankel matrix is an infinite matrix F each of its rows and columns are indexed by strings in Σ^*.
- The (x, y) entry of F ($F_{x,y}$) contains the value $f(xy)$.

Example (a-count function)

$$F = \begin{pmatrix} & \epsilon & a & b & aa & ab & ba & bb & \dots \\ \epsilon & 0 & 1 & 0 & 2 & 1 & 1 & 0 & \dots \\ a & 1 & 2 & 1 & 3 & 2 & 2 & 1 & \dots \\ b & 0 & 1 & 0 & 2 & 1 & 1 & 0 & \dots \\ aa & 2 & 3 & 2 & 4 & 3 & 3 & 2 & \dots \\ ab & 1 & 2 & 1 & 3 & 2 & 2 & 1 & \dots \\ ba & 1 & 2 & 1 & 3 & 2 & 2 & 1 & \dots \\ bb & 0 & 1 & 0 & 2 & 1 & 1 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Theorem (Carlyle and Paz theorem, 1971)

Let $f : \Sigma^* \to \mathcal{K}$ such that $f \not\equiv 0$ and let F be the corresponding Hankel matrix. Then, the size r of the smallest MA A such that $f_A \equiv f$ satisfies $r = \mathrm{rank}(F)$ (over the field)

Example (a-count function)

The rank is 2, F_ϵ and F_a are a basis.
The other rows:

$$F_\epsilon = (1,0)(F_\epsilon, F_a)^t \qquad F_a = (0,1)(F_\epsilon, F_a)^t$$
$$F_b = (1,0)(F_\epsilon, F_a)^t \qquad F_{aa} = (-1,2)(F_\epsilon, F_a)^t$$
$$F_{ab} = (0,1)(F_\epsilon, F_a)^t \qquad F_{ba} = (0,1)(F_\epsilon, F_a)^t$$
$$F_{bb} = (1,0)(F_\epsilon, F_a)^t \qquad \ldots$$

Theorem (Carlyle and Paz theorem, 1971)

Let $f : \Sigma^* \rightarrow \mathcal{K}$ such that $f \not\equiv 0$ and let F be the corresponding Hankel matrix. Then, the size r of the smallest MA A such that $f_A \equiv f$ satisfies $r = \mathrm{rank}(F)$ (over the field)

Example (a-count function)

The rank is 2, F_ϵ and F_a are a basis.
The other rows:

$$
\begin{aligned}
F_\epsilon &= (1,0)(F_\epsilon, F_a)^t & F_a &= (0,1)(F_\epsilon, F_a)^t \\
F_b &= (1,0)(F_\epsilon, F_a)^t & F_{aa} &= (-1,2)(F_\epsilon, F_a)^t \\
F_{ab} &= (0,1)(F_\epsilon, F_a)^t & F_{ba} &= (0,1)(F_\epsilon, F_a)^t \\
F_{bb} &= (1,0)(F_\epsilon, F_a)^t & & \dots
\end{aligned}
$$

Carlyle and Paz theorem [1971]

Note:

Let $x_1 = \epsilon, x_2, \ldots, x_r$ a basis of the Hankel matrix. The Theorem states that we can build the MA as:

- $\lambda = (1, 0, \ldots, 0)$; $\gamma = (f(x_1), \ldots, f(x_r))$
- for every σ, define the ith row of the matrix μ_σ as the (unique) coefficients of the row $F_{x_i\sigma}$ when expressed as a linear combination of $F_{x_1}, \ldots, F_{x_r}$. That is:

$$F_{x_i\sigma} = \sum_{j=1}^{r} [\mu_\sigma]_{i,j} F_{x_j}$$

Example (a-count function)

$$\gamma = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad \mu_a = \begin{pmatrix} F_{\epsilon \cdot a} \\ F_{a \cdot a} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix} \quad \mu_b = \begin{pmatrix} F_{\epsilon \cdot b} \\ F_{a \cdot b} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Carlyle and Paz theorem [1971]

Note:

Let $x_1 = \epsilon, x_2, \ldots, x_r$ a basis of the Hankel matrix. The Theorem states that we can build the MA as:

- $\lambda = (1, 0, \ldots, 0)$; $\gamma = (f(x_1), \ldots, f(x_r))$
- for every σ, define the ith row of the matrix μ_σ as the (unique) coefficients of the row $F_{x_i\sigma}$ when expressed as a linear combination of $F_{x_1}, \ldots, F_{x_r}$. That is:

$$F_{x_i\sigma} = \sum_{j=1}^{r} [\mu_\sigma]_{i,j} F_{x_j}$$

Example (a-count function)

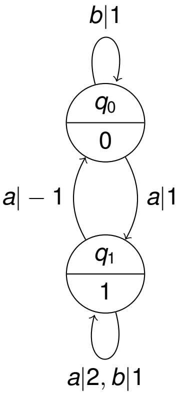
$$\gamma = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad \mu_a = \begin{pmatrix} F_{\epsilon \cdot a} \\ F_{a \cdot a} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix} \quad \mu_b = \begin{pmatrix} F_{\epsilon \cdot b} \\ F_{a \cdot b} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Note:

Let $x_1 = \epsilon, x_2, \ldots, x_r$ a basis of the Hankel matrix. The Theorem states that we can build the MA as:

- $\lambda = (1, 0, \ldots, 0)$; $\gamma = (f(x_1), \ldots, f(x_r))$
- for every σ, define the ith row of the matrix μ_σ as the (unique) coefficients of the row $F_{x_i\sigma}$ when expressed as a linear combination of $F_{x_1}, \ldots, F_{x_r}$. That is:

$$F_{x_i\sigma} = \sum_{j=1}^{r} [\mu_\sigma]_{i,j} F_{x_j}$$



Example (a-count function)

$$\gamma = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad \mu_a = \begin{pmatrix} F_{\epsilon \cdot a} \\ F_{a \cdot a} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix} \quad \mu_b = \begin{pmatrix} F_{\epsilon \cdot b} \\ F_{a \cdot b} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Outline

1. Multiplicity Automata

2. Exact Learning

3. A bit of Algebra

4. Examples

Definition (Equivalence query)

Let f be a target function.
Given a hypothesis h, an equivalence query $(\mathrm{EQ}(h))$ returns:

- **YES** if $h \equiv f$
- a counterexample otherwise

Definition (Membership query)

Let f be a target function.
Given an assignment z a membership query $(\mathrm{MQ}(z))$ returns $f(z)$

Definition (Equivalence query)

Let f be a target function.
Given a hypothesis h, an equivalence query $(\mathrm{EQ}(h))$ returns:

- **YES** if $h \equiv f$
- a counterexample otherwise

Definition (Membership query)

Let f be a target function.
Given an assignment z a membership query $(\mathrm{MQ}(z))$ returns $f(z)$

Definition (Angluin, 1988)

Given a target function f, a learning algorithm should return a hypothesis function h equivalent to f.
In order to do so, the learner can resort to membership and equivalence queries.
We say that the learner learns a class of functions $\mathcal{C}$, if, for every function $f \in \mathcal{C}$, the learner outputs a hypothesis h that is equivalent to f and does so in time polynomial in the "size" of a shortest representation of f and the length of the longest counterexample.

The idea is to work with a finite version of the Hankel matrix.

Algorithm

1. initialize the matrix to null
2. build a MA using the matrix and making membership queries if necessary
3. ask an equivalence query
4. if the answer is **YES** then STOP
5. use the counterexample to add new rows an columns in the matrix
6. use membership queries to fill the holes in the matrix
7. Go to step 2

Outline

1. Multiplicity Automata

2. Exact Learning

3. A bit of Algebra

4. Examples

Definition (Field)

$(\mathcal{K}, +, *)$ is a field if:

- Closure of $\mathcal{K}$ under $+$ and $*$ $\forall a, b \in \mathcal{K}$, both $a + b$ and $a * b$ belong to $\mathcal{K}$
- Both $+$ and $*$ are associative $\forall a, b, c \in \mathcal{K}$, $a + (b + c) = (a + b) + c$ and $a * (b * c) = (a * b) * c$.
- Both $+$ and $*$ are commutative $\forall a, b \in \mathcal{K}$, $a + b = b + a$ and $a * b = b * a$.
- The operation $*$ is distributive over the operation $+$ $\forall a, b, c \in \mathcal{K}$, $a * (b + c) = (a * b) + (a * c)$.
- Existence of an additive identity $\exists 0 \in \mathcal{K}$: $\forall a \in \mathcal{K}$, $a + 0 = a$.
- Existence of a multiplicative identity $\exists 1 \in \mathcal{K}$, $1 \neq 0$: $\forall a \in \mathcal{K}$, $a * 1 = a$.
- Existence of additive inverses $\forall a \in \mathcal{K}$, $\exists -a \in \mathcal{K}$: $a + (-a) = 0$.
- Existence of multiplicative inverses $\forall a \in \mathcal{K}$, $a \neq 0$, $\exists a^{-1} \in \mathcal{K}$: $a * a^{-1} = 1$.

Idea:

Use the learning algorithm using:

- concatenation as the $*$ operator
- the inclusion in a (multi)set as the $+$ operator

We are going to extend this operations in order to have a Field and be able to identify a superclass of the ambiguous rational transducers

Concatenation extension

- The concatenation is going to play the role of the multiplication.
- For each $a \in \Sigma$ let we include in Σ its inverse (a^{-1}).

Example

$$aabb \qquad aba^{-1}b$$

$$aaa^{-1}b \quad (\equiv ab) \qquad a^{-1}b^{-1}$$

Extended concatenation properties:

- Closure
- Associative
- Non Commutative (not good)
- Existence of a multiplicative identity (ϵ)
- Existence of multiplicative inverses

Concatenation extension

- The concatenation is going to play the role of the multiplication.
- For each $a \in \Sigma$ let we include in Σ its inverse (a^{-1}).

Example

$$aabb \qquad aba^{-1}b$$

$$aaa^{-1}b \quad (\equiv ab) \qquad a^{-1}b^{-1}$$

Extended concatenation properties:

- Closure
- Associative
- Non Commutative (not good)
- Existence of a multiplicative identity (ϵ)
- Existence of multiplicative inverses

- The concatenation is going to play the role of the multiplication.
- For each $a \in \Sigma$ let we include in Σ its inverse (a^{-1}).

Example

$$aabb \qquad aba^{-1}b$$
$$aaa^{-1}b \quad (\equiv ab) \qquad a^{-1}b^{-1}$$

Extended concatenation properties:

- Closure
- Associative
- Non Commutative (not good)
- Existence of a multiplicative identity (ϵ)
- Existence of multiplicative inverses

- the multiset inclusion is going to play the role the addition
- For each multiset x let we define its inverse $(-x)$.

Example

$$aaa + bbb \qquad aaa - aaa \quad (\equiv \emptyset)$$

$$a + a - a \quad (\equiv a) \qquad -aaa$$

Multiset inclusion properties:

- Closure: the inclusion of a multiset into another is a multiset.
- Associative: $(x + y) + z = x + (y + z)$
- Commutative: $x + y = y + x$
- Existence of an additive identity: $x + \emptyset = x$
- Existence of additive inverses: $x + (-x) = \emptyset$

Multiset inclusion extension

- the multiset inclusion is going to play the role the addition
- For each multiset x let we define its inverse $(-x)$.

Example

$$aaa + bbb \qquad aaa - aaa \quad (\equiv \emptyset)$$

$$a + a - a \quad (\equiv a) \qquad -aaa$$

Multiset inclusion properties:

- Closure: the inclusion of a multiset into another is a multiset.
- Associative: $(x + y) + z = x + (y + z)$
- Commutative: $x + y = y + x$
- Existence of an additive identity: $x + \emptyset = x$
- Existence of additive inverses: $x + (-x) = \emptyset$

- the multiset inclusion is going to play the role the addition
- For each multiset x let we define its inverse $(-x)$.

Example

$$aaa + bbb \qquad aaa - aaa \quad (\equiv \emptyset)$$

$$a + a - a \quad (\equiv a) \qquad -aaa$$

Multiset inclusion properties:

- Closure: the inclusion of a multiset into another is a multiset.
- Associative: $(x + y) + z = x + (y + z)$
- Commutative: $x + y = y + x$
- Existence of an additive identity: $x + \emptyset = x$
- Existence of additive inverses: $x + (-x) = \emptyset$

Properties:

- The concatenation is distributive over the inclusion:
  $x * (y + z) = x * y + x * z$

- We have a “Field” with a non commutative multiplication.
- This is known as a Divisive Ring
- But the Carlyle an Paz theorem does not use the commutativity in the multiplication!
- Their theorem is also true for Divisive Rings!
- Then the inference algorithm can be used exactly as it is just substituting:
  - addition by the (extended) inclusion
  - multiplication by the (extended) concatenation

Properties:

- The concatenation is distributive over the inclusion:
  $x * (y + z) = x * y + x * z$
- We have a “Field” with a non commutative multiplication.
- This is known as a Divisive Ring
- But the Carlyle an Paz theorem does not use the commutativity in the multiplication!
- Their theorem is also true for Divisive Rings!
- Then the inference algorithm can be used exactly as it is just substituting:
  - addition by the (extended) inclusion
  - multiplication by the (extended) concatenation

Outline

1. Multiplicity Automata

2. Exact Learning

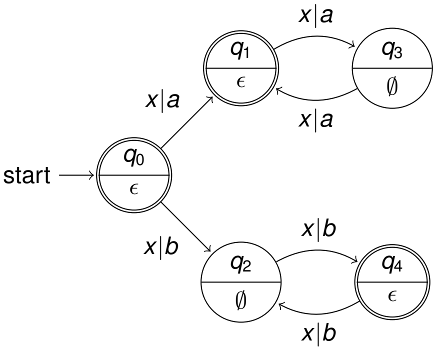
3. A bit of Algebra

4. Examples

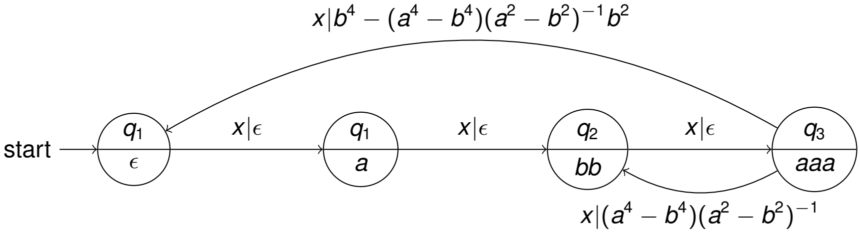
Non ambiguous, non subsequential example

$$f(x^n) = \begin{cases} a^n & \text{if } n \text{ is odd} \\ b^n & \text{if } n \text{ is even} \end{cases}$$

Text books proposal:



Applying the algorithm we obtain:

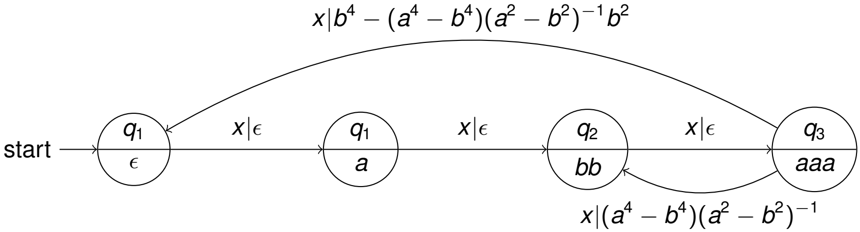


- It can be shown that for any input we obtain a plain string

Open questions:

- Does there exist a general method to simplify and compare string expressions?
- Does there exist a method to know if a multiplicity automaton produces only plain strings?
- Does there exist a method to remove complex expressions in arcs and states, possibly adding more states?

Applying the algorithm we obtain:



- It can be shown that for any input we obtain a plain string

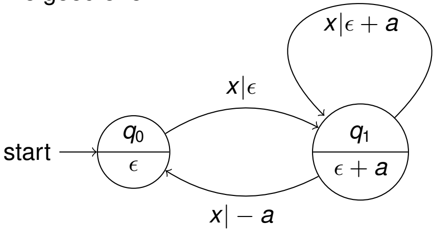
Open questions:

- Does there exist a general method to simplify and compare string expressions?
- Does there exist a method to know if a multiplicity automaton produces only plain strings?
- Does there exist a method to remove complex expressions in arcs and states, possibly adding more states?

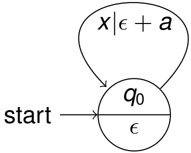
An ambiguous example

$$f(x^n) = \sum_{i=0}^{n} a^i$$

The good one



A non equivalent one



- The second transducer does not preserve the multiplicity of the strings
- Note that in the ambiguous case, the membership query should return all the possible transductions.

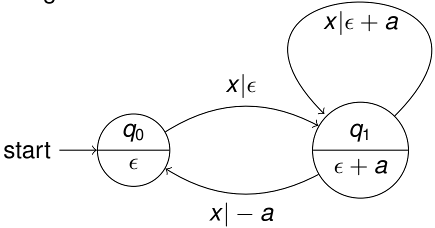
Open questions:

- Can we still be able to learn if only information about just one transduction is provided in each query?
- Does any learnable function remain learnable if the multiplicity is not taken into account?

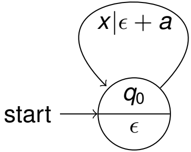
An ambiguous example

$$f(x^n) = \sum_{i=0}^{n} a^i$$

The good one



A non equivalent one



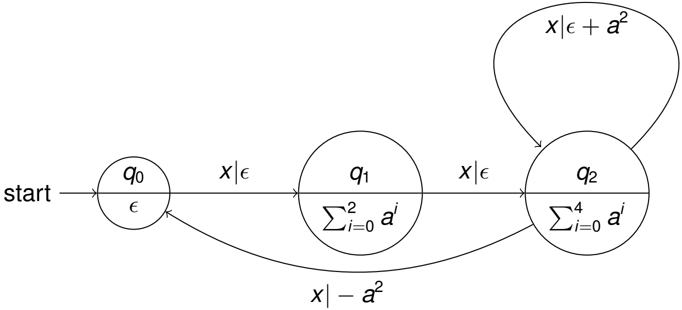
- The second transducer does not preserve the multiplicity of the strings
- Note that in the ambiguous case, the membership query should return all the possible transductions.

Open questions:

- Can we still be able to learn if only information about just one transduction is provided in each query?
- Does any learnable function remain learnable if the multiplicity is not taken into account?

$$f(x^n) = \sum_{i=0}^{2n} a^i$$

Applying the algorithm:



We have proposed a learning algorithm that:

- Can identify any rational fuction with output built up with
  - no empty-transitions
  - extended concatenations
  - extended multiset inclusions
- It uses membership and equivalence queries
- As a special case, it identifies any ambiguous rational transducer (with finite output)
- It works in polynomial time (perhaps there is a problem in the parsing)

Any Questions?