

# Implementation of transducers in Vaucanson

Sarah O'Connor <[sarah.o-connor@lrde.epita.fr](mailto:sarah.o-connor@lrde.epita.fr)>

LRDE seminar, June 23, 2004

<http://vaucanson.lrde.epita.fr/>



## Copying this document

Copyright © 2004 LRDE.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with the Invariant Sections being just “Copying this document”, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is provided in the file COPYING.DOC.

# Introduction

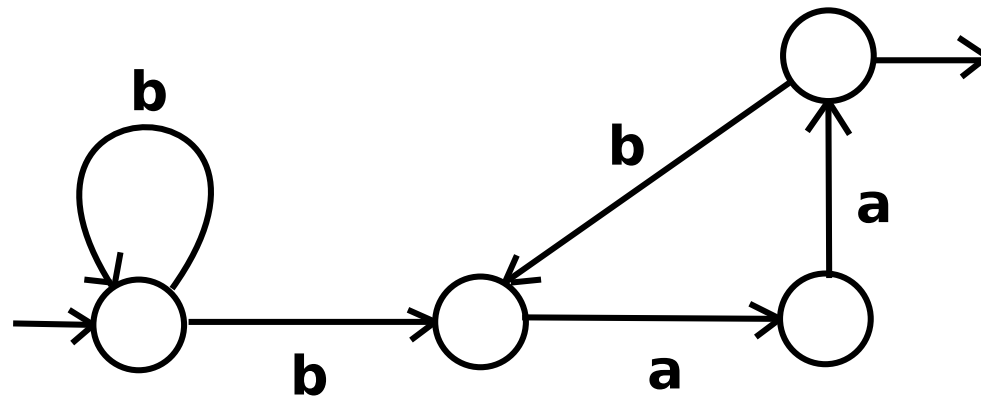
- What is Vaucanson?
  - generic library
  - automata manipulation
  
- What are we going to talk about?
  - transducers
  - implementation in Vaucanson

# Table of Contents

<b>Introduction</b> .....	<b>2</b>
<b>Theory</b> .....	<b>6</b>
<b>Implementation</b> .....	<b>16</b>
<b>Conclusion</b> .....	<b>25</b>

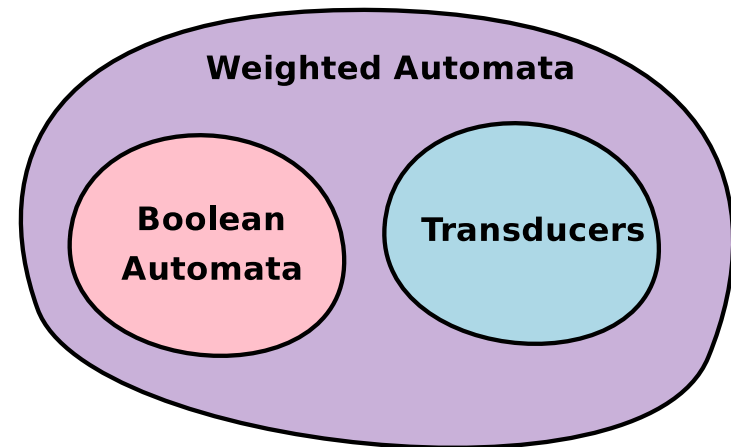
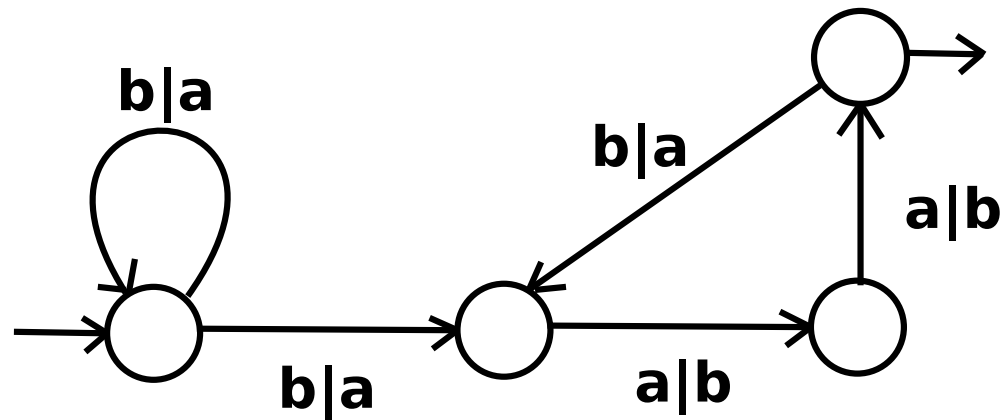
## Preliminaries (1/2)

What is an automaton?



## Preliminaries (2/2)

What is a transducer?



# Theory

## Algebraic structures (1/4)

### Monoid and Free monoid

A **monoid**  $M$  is a set with:(3)

- An associative binary operation  $(\cdot)$ .
- A neutral element  $(1_M)$ .

Word = list of letters of an alphabet  $A$ .

$A^*$  = set of words built on  $A$ .

$A^*$  is a **free monoid** (op = concatenation, neutral element = empty word).

## Algebraic structures (2/4)

### Freemonoid Product

Cartesian product of two free monoids:

$$w_1 \in A^*, w_2 \in B^* \Rightarrow (w_1, w_2) \in A^* \times B^*$$

Remarks:

- Free monoid product = monoid but  $\neq$  free monoid.

$$\text{Example: } (a, b) = \begin{cases} (a, 1) \cdot (1, b) \\ (1, b) \cdot (a, 1) \end{cases}$$

- Can be extended to  $n$  free monoids.

## Algebraic structures (3/4)

### Example

- Alphabets:  $A = \{a, b, c\}$  ,  $B = \{x, y, z\}$ .
- Free monoids:
  - $A^*$   $\rightarrow$  words built on  $A$ ,
  - $B^*$   $\rightarrow$  words built on  $B$ .
- Free monoid product ( $A^* \times B^*$ )
  - (ab, x)
  - (abba, zz)
  - ...

## Algebraic structures (4/4)

### Semiring

A semiring  $\mathbb{K}$  is a set with two binary operations: sum (+) and product ( $\cdot$ ). (3)

- $(\mathbb{K}, +)$  : commutative monoid, neutral element :  $0_{\mathbb{K}}$ .
- $(\mathbb{K}, \cdot)$  : monoid, neutral element :  $1_{\mathbb{K}}$ .
- Product is distributive on sum.
- $0_{\mathbb{K}}$  is a zero for product.

## Series and Automata

- A **series** is an application from a monoid to a semiring.
- The set of series on the monoid  $M$  in the semiring  $\mathbb{K}$  is:  $\mathbb{K}\langle\langle M \rangle\rangle$ .

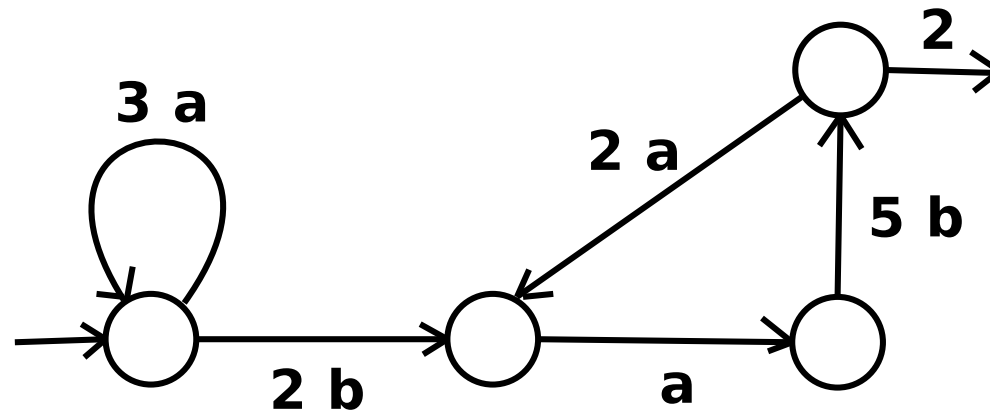
Example: a Boolean series represents a set of words (= a language).

$s = a \cdot (b+c)$  recognizes the language  $\{ab, ac\}$ .

An **automaton** is in fact a series.

## Weighted Automata

In the case of **weighted automata**, a word is no longer accepted or refused, it is “evaluated”.



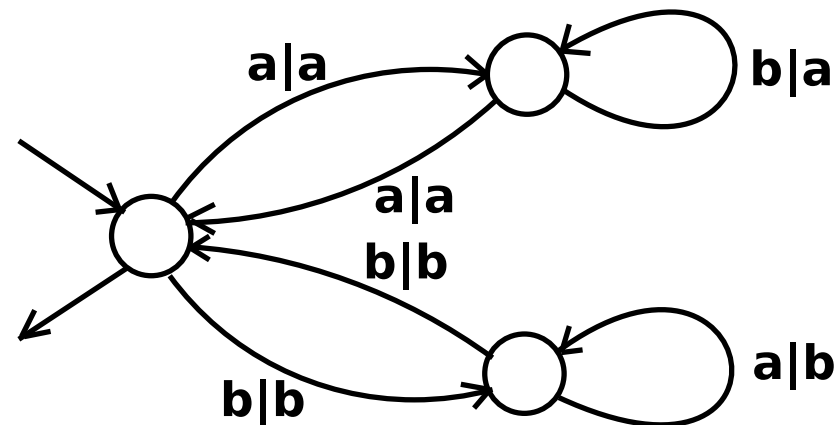
On semiring  $(\mathbb{R}, +, \times)$ , evaluation of the word **aabab**:

$$120 = 3 \times 3 \times 2 \times 1 \times 5 \times 2$$

# Transducers

- set of series = semiring
- transducer = weighted automaton

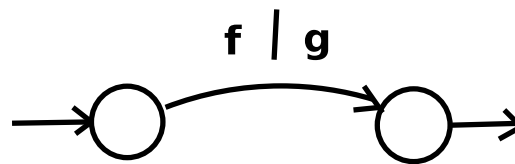
⇒ A **transducer** is an automaton built on such a semiring. It associates a **set of words** to the word given in input.



## Swapping between monoids

Two different views:(3)

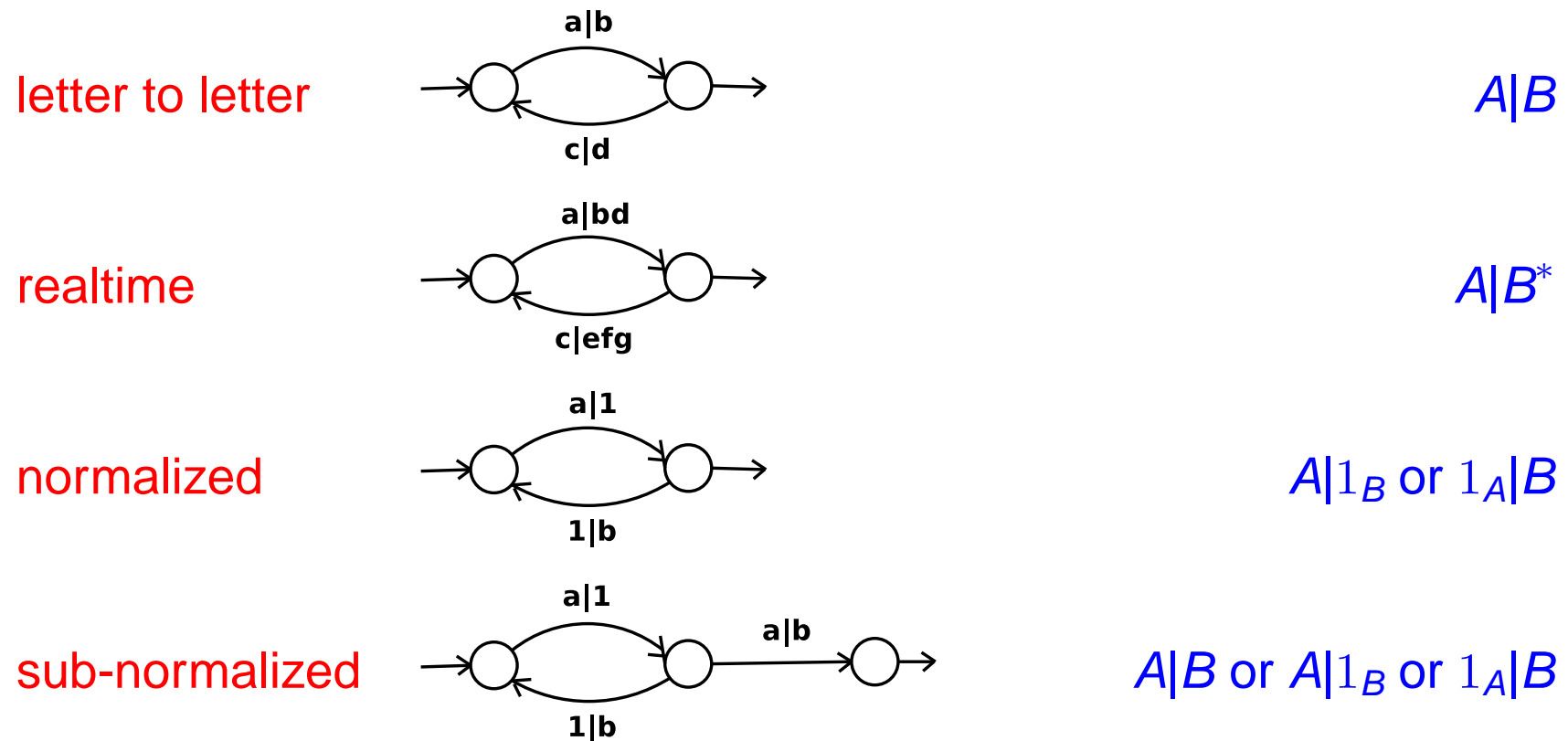
- Weighted automaton on  $A^*$  with multiplicity in  $\mathbb{B} \langle\langle B^* \rangle\rangle$ :  $(\mathbb{B} \langle\langle B^* \rangle\rangle) \langle\langle A^* \rangle\rangle$ .
- Automaton on the monoid  $A^* \times B^*$ .



- $f \in A^*, g \in \mathbb{B} \langle\langle B^* \rangle\rangle \Rightarrow$  in:  $f$ , out:  $g$
- $f \in A^*, g \in B^* \Rightarrow$  in:  $(f, g)$ , out: true

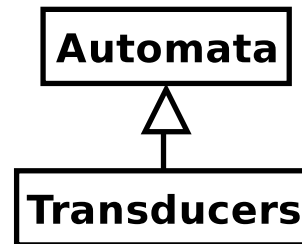
## Special transducers

There are **categories** in which a transducer can be classified:



# Implementation

## Transducer class



**Why?**

**Because:**

- is a weighted automaton,
- better typing,
- extended services.

## Transducers on $(\mathbb{B} \langle\langle \mathbf{B}^* \rangle\rangle) \langle\langle \mathbf{A}^* \rangle\rangle$

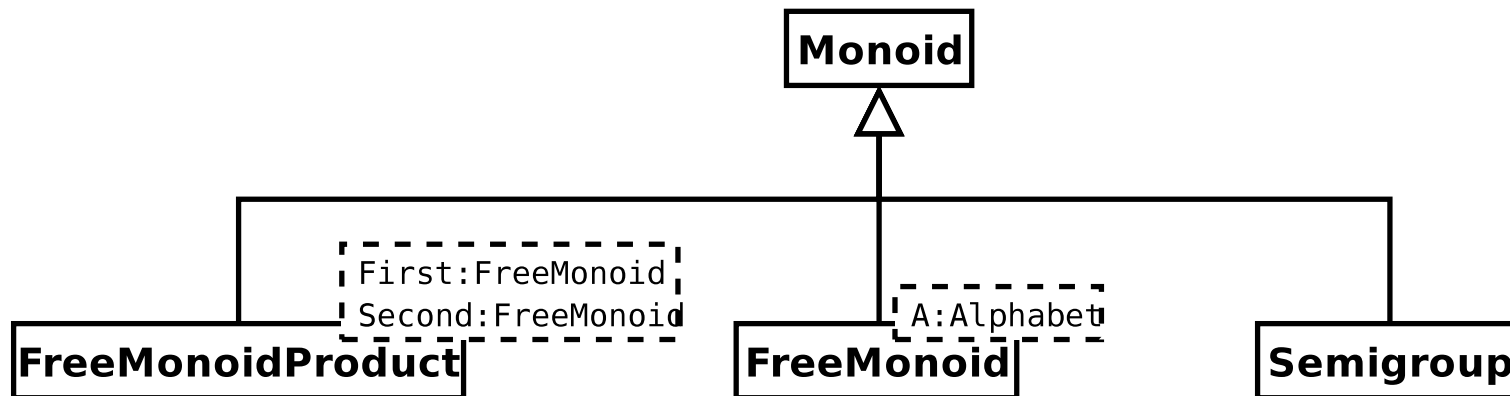
Already **existed** in Vaucanson, but needed to be **tested**.

- Evaluation
- Recognition for some categories
- Composition for some categories

Add sugar to facilitate use.

# Automata on $A^* \times B^*$

- The notion of **freemonoid product** had to be included in VAUCANSON .(LRDE)



- Services are being provided. (length, swap...)

## Instanciating a transducer on $A^* \times B^*$

- Why a transducer?
  - Although **automaton** on  $A^* \times B^* \Rightarrow$  **transducer**.
  - Class Transducers inherits from class Automata  $\Rightarrow$  no loss of information.
- Design pattern Element (1)
  - `Element<FreeMonoidProduct, T>`
  - `T = std::pair<std::string, std::string>`
- **Problem** encountered.

## Understanding the problem

In class AutomataBase:(LRDE)

```
/** type of the free monoid. */  
typedef typename series_set_t::monoid_t           monoid_t;  
  
/** type of the letter. */  
typedef typename monoid_t::letter_t             letter_t;
```

Here, monoid\_t = **FreeMonoidProduct**.

⇒ **letter\_t** makes no sense and does not exist!

## Attempting to solve the problem

Problem: no `alphabet_t` and no `letter_t`.

⇒ **Intermediary solution** for the moment:

In class `FreeMonoidProduct`,

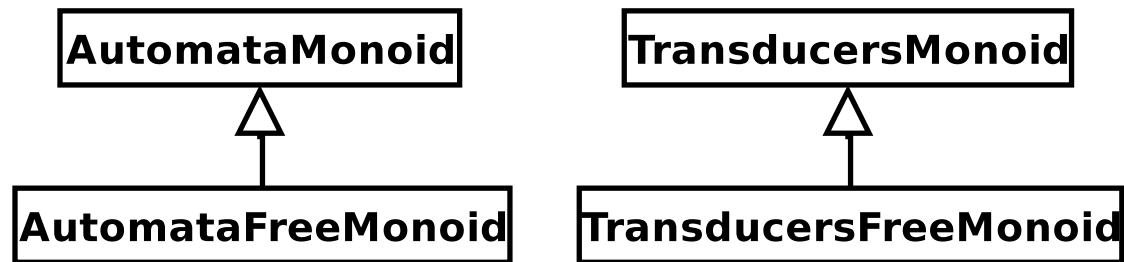
```
typedef undefined_type    alphabet_t;  
typedef undefined_type    letter_t;
```

⇒ Intermediary because:

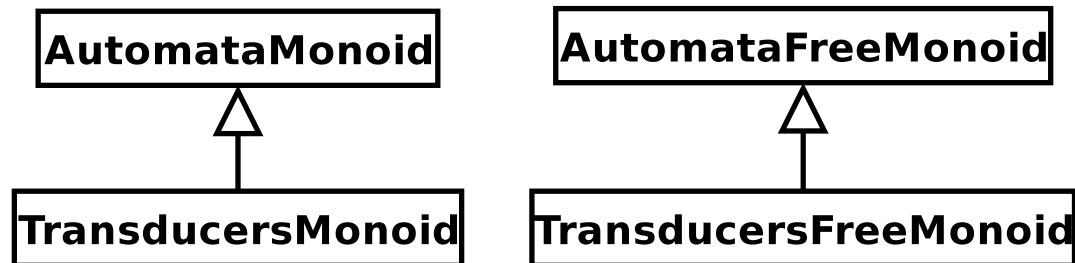
- not **safe**: **simulating** the existence of something → never good.
- not **satisfying**: **deeper** solution needed.

## Towards a better solution

We would like:



But also:



## Further work

⇒ Algorithms on **transducers**:

- swap between monoids,
- transformation to sub-normalized.

⇒ Remove **intermediary solution**:

- remove typedefs,
- change the hierarchy.

⇒ Still lots of work to do:

- algorithms on categories of transducers.
- small algorithms on transducers over a product of free monoids (longest common prefix...)

## Conclusion

- Already existing **algorithms**:
  - tests have been written,
  - can be used in programs.
- **FreeMonoidProduct**:
  - added to Vaucanson,
  - totally usable on its own,
  - a few more services needed.
- **Transducers** on FreeMonoidProduct
  - works thanks to an intermediary solution,
  - work in progress: researching a better model.

## References

- [1] Lombardy, S., Poss, R., Régis-Gianas, Y., and Sakarovitch, J. (2003).  
Introducing VAUCANSON.
  
- [2] O'Connor, S. (2004). Implementation of transducers in VAUCANSON.  
Technical report, LRDE.
  
- [3] Sakarovitch, J. (2003). *Éléments de théorie des automates*. Vuibert  
Informatique.
  
- [LRDE] LRDE. VAUCANSON. <http://vaucanson.lrde.epita.fr/>.