Inside VAUCANSON

The VAUCANSON group

LRDE / EPITA - LIAFA / Paris 7 - LTCI / ENST

June 27, 2005

Aim of this talk

About the VAUCANSON platform:

- starts to be usable, recently released version 0.7.1,
- new algorithms have been implemented,
- new organization of the library.

Overview of VAUCANSON features

VAUCANSON deals with:

- automata with any multiplicity ($\mathbb{B}$, $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{Q}$, ...),
- rational expressions (including weighted expressions),
- transducers.

VAUCANSON Input / Output:

- FSM library,
- XML.

Overview of VAUCANSON features

VAUCANSON deals with:

- automata with any multiplicity ($\mathbb{B}$, $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{Q}$, $\ldots$),
- rational expressions (including weighted expressions),
- transducers.

VAUCANSON Input / Output:

- FSM library,
- XML.

Some caracteristic features

Programming concerns:

- all components are generic,
- context headers (i.e. predefined types) are introduced.

⇒ Library manipulation easiness is improved.

Some predefined types:

- Boolean automaton, weighted automaton ($\mathbb{R}$, tropical, ...),
- transducers.

Some caracteristic features

Programming concerns:

- all components are generic,
- context headers (i.e. predefined types) are introduced.

$\Rightarrow$ Library manipulation easiness is improved.

Some predefined types:

- Boolean automaton, weighted automaton ($\mathbb{R}$, tropical, ...),
- transducers.

Some caracteristic features

Programming concerns:

- all components are generic,
- context headers (i.e. predefined types) are introduced.

$\Rightarrow$ Library manipulation easiness is improved.

Some predefined types:

- Boolean automaton, weighted automaton ($\mathbb{R}$, tropical, ...),
- transducers.

Coding the B_1 automaton

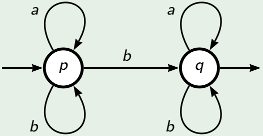
Example

```
#include <vaucanson/boolean_automaton.hh>
using namespace vcsn;
using namespace vcsn::boolean_automaton;

int main() {
  alphabet_t alpha; alpha.insert('a'); alpha.insert('b');
  automaton_t B1 = new_automaton(alpha);

  hstate_t p = B1.add_state();
  hstate_t q = B1.add_state();
  B1.set_initial(p);
  B1.set_final(q);

  B1.add_letter_edge(p, p, 'a');
  B1.add_letter_edge(p, p, 'b');
  B1.add_letter_edge(q, q, 'a');
  B1.add_letter_edge(q, q, 'b');
  B1.add_letter_edge(p, q, 'b');
}
```



n^{th} power of B_1

Compute the n^{th} power of B_1:

Example

```
automaton_t p = a;
for (int i = 1; i < n; ++i)
  p = product(p, a);
```

B_1 seen as a $\mathbb{Z}$-automaton

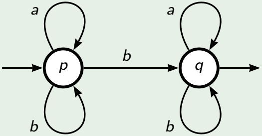
Example

```
#include <vaucanson/z_automaton.hh>
using namespace vcsn;
using namespace vcsn::z_automaton;

int main() {
  alphabet_t alpha; alpha.insert('a'); alpha.insert('b');
  automaton_t BZ1 = new_automaton(alpha);

  hstate_t p = BZ1.add_state();
  hstate_t q = BZ1.add_state();
  BZ1.set_initial(p);
  BZ1.set_final(q);

  BZ1.add_letter_edge(p, p, 'a');
  BZ1.add_letter_edge(p, p, 'b');
  BZ1.add_letter_edge(q, q, 'a');
  BZ1.add_letter_edge(q, q, 'b');
  BZ1.add_letter_edge(p, q, 'b');
}
```



Algorithms in VAUCANSON

- Over than 50 algorithms are available,
- implementation supports weights whenever it is possible.

Some special algorithms:

- computation of the minimal quotient,
- partial derivatives automaton,
- ...

Partial derivatives

- generalization of Antimirov's method to weighted expressions,
- extend implementation proposed on unweighted expressions by Champarnaud and Ziadi,
- complexity is likely to be quadratic,
- Derived term automaton is a quotient of Glushkov(E).

Some results

Experiment: let $\mathcal{A}_{15}$ be an automaton with 15 states, provided its determinist equivalent has 2^{15} states. We compute expressions from the automaton, with a random chooser on state ordering.

Class	l_E	Derived term $\mathcal{A}_E$		Standard $\mathcal{S}_E$	
		$\mathcal{A}_E$ states	time	$\mathcal{V}_E$ states	time
1	110	24	0.123	24	0.012
7	410	53	0.470	51	0.050
14	1035	66	1.169	60	0.138
20	7821	90	13.412	78	1.418

Results enlights:

- partial derivative algorithm gives smaller results

but:

- is slower than $Quotient(Glushkov(E))$.

Reminder on transducers

A transducer is an automaton labeled by pair of words.

Theorem

- *A transducer can be normalized.*
- *A transducer can be put in the form of an automaton on A^* with weights in $Rat(B^*)$.*

In VAUCANSON, both transducer forms are available.

About composition in VAUCANSON

Two composition algorithms available:

- composition of transducers with weights in $Rat(B^*)$ (Schützenberger 61),
- composition of (sub-) normalized transducers.

Composition of unweighted transducers

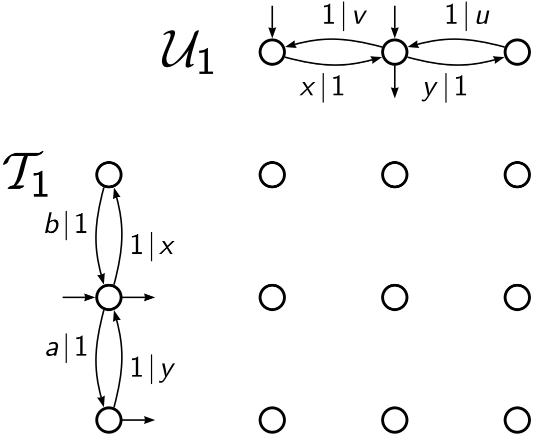


Figure: Composition Theorem on Boolean transducers

Composition of unweighted transducers

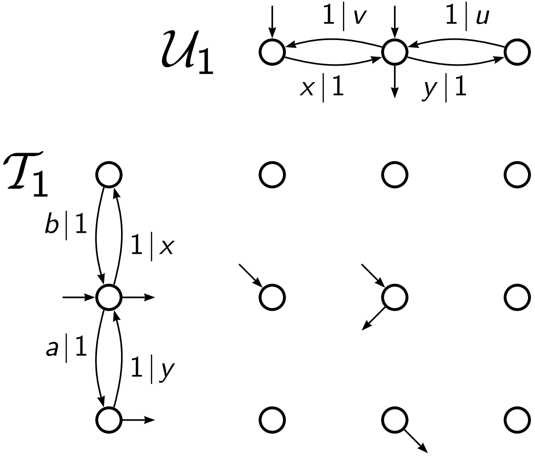


Figure: Composition Theorem on Boolean transducers

Composition of unweighted transducers

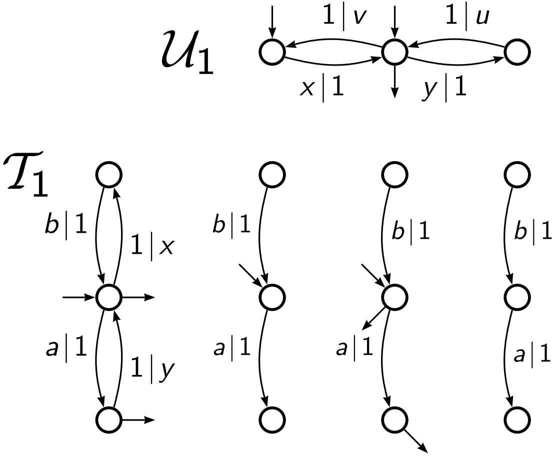


Figure: Composition Theorem on Boolean transducers

Composition of unweighted transducers

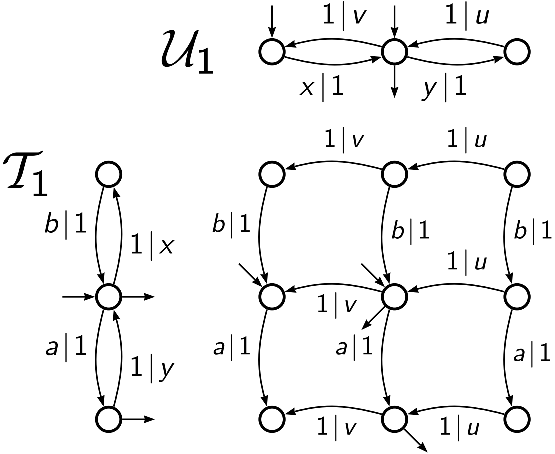


Figure: Composition Theorem on Boolean transducers

Composition of unweighted transducers

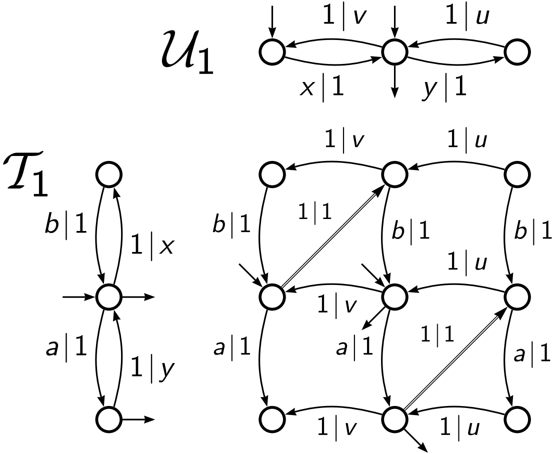


Figure: Composition Theorem on Boolean transducers

Composition of unweighted transducers

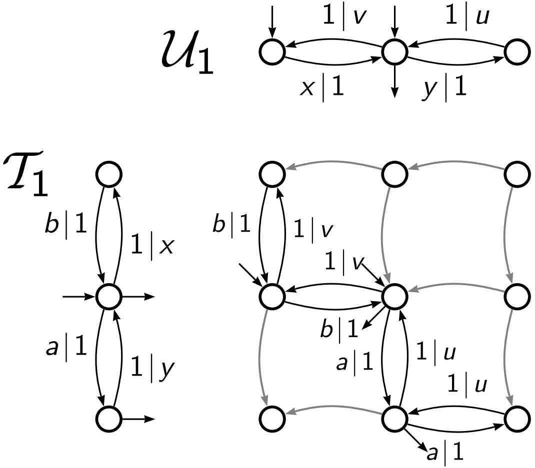


Figure: Composition Theorem on Boolean transducers

Composition of weighted transducers

Two possible solutions:

- instanciation of an empty word and apply a filter *after* the composition (Mohri, Pereira, Riley 96-00),
- modify the composed transducers *before* the composition.

Proposed algorithm for weighted transducers

Principle: Modify the input transducers and suppress the problematic states after the composition.

Algorithm

- *Out-splitting on first transducer: separate states which have empty outputs,*
- *in-splitting on second transducer: separate states which have empty inputs,*
- *mark problematic states,*
- *compose transducers,*
- *suppress intersection of marked states.*

Out-splitting

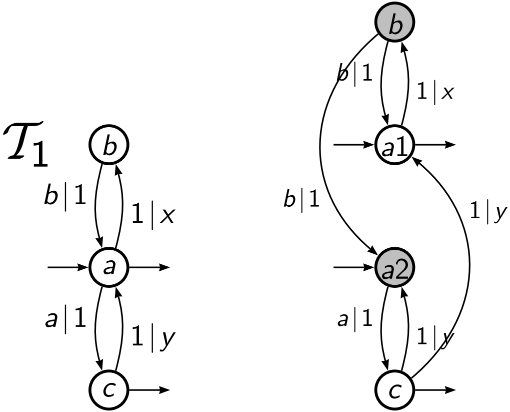


Figure: Out-splitting

In-splitting

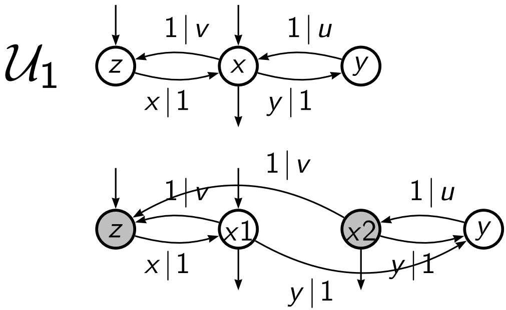
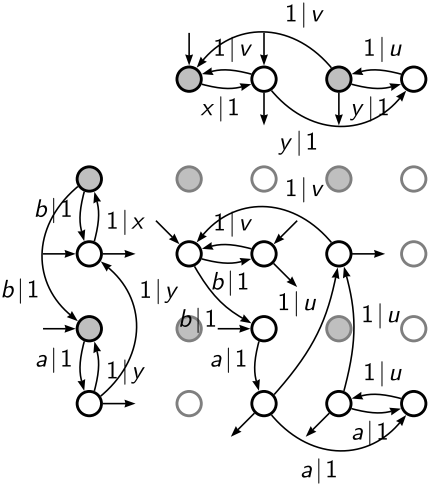


Figure: In-splitting

A composition that preserves multiplicity



Some results

Composition of the left sequential transducer and the right sequential transducer for the rewriting rule $ab^n \longrightarrow ba^n$:

Algorithm	n	Nb. states	Nb. transitions	Time
Sub-normalized transducer	20	30084	40356	0.551
	40	232564	305506	4.849
Representation	20	441	882	2.042
	40	1681	3362	36.195

Summary

VAUCANSON is a `generic` framework for automata manipulation.

The library offers several services on:

- various automaton types,
- rational expressions,
- various transducer types.

Get VAUCANSON: http://vaucanson.lrde.epita.fr

Questions