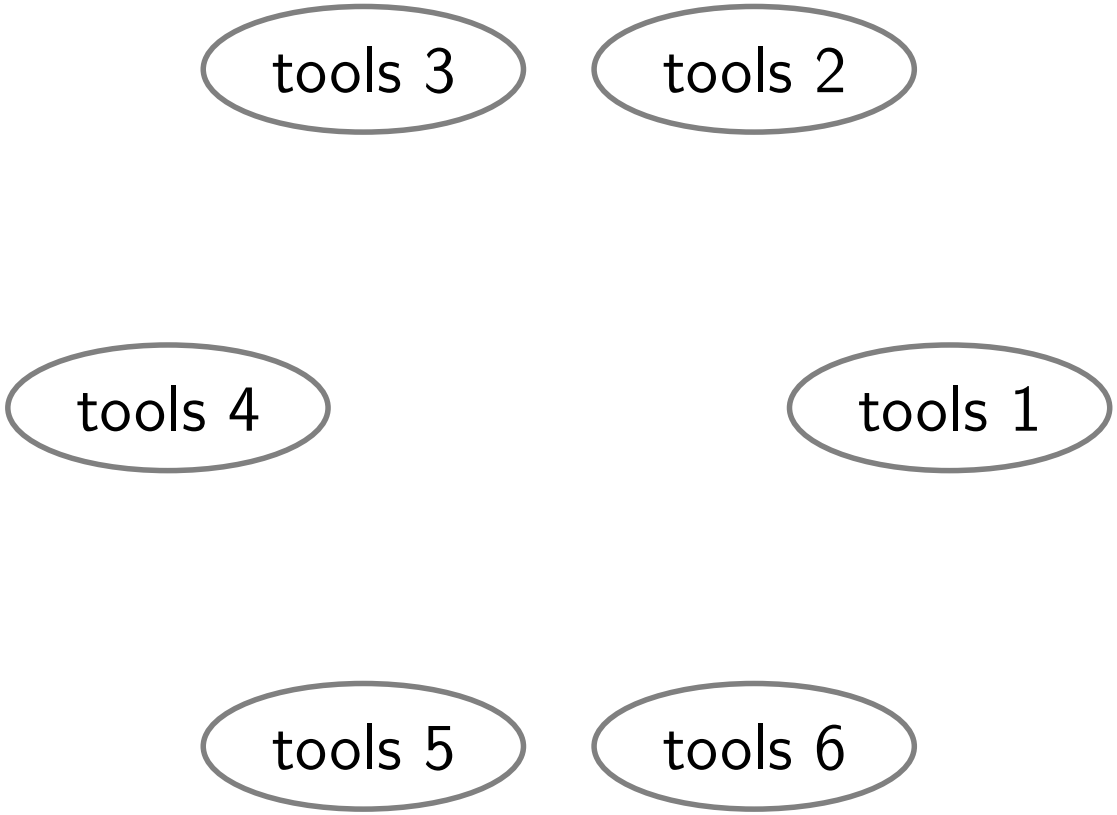
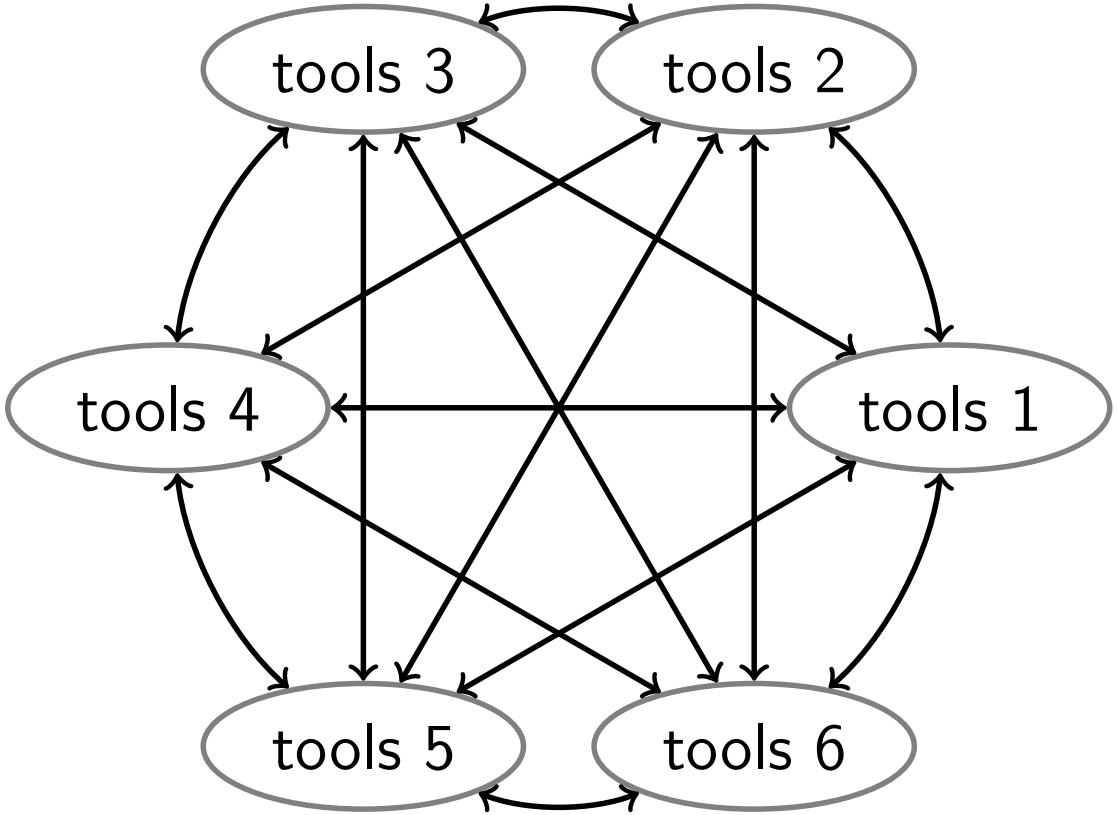
An XML format for the description of weighted automata, transducers, and regular expressions

A. Demaille A. Duret-Lutz F. Lesaint S. Lombardy
J. Sakarovitch F. Terrones

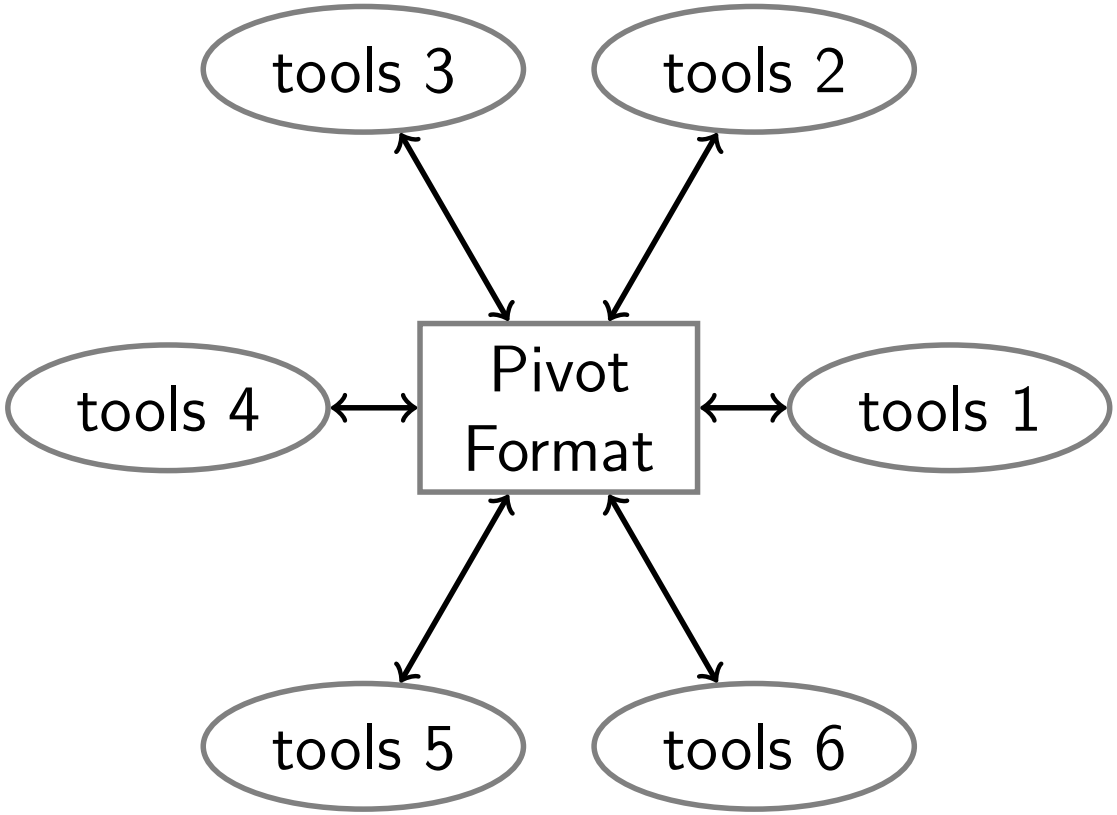
The problem we address



The problem we address



The problem we address



An exchange format that:

- stores descriptions of any kind of automata,
- allows an easy implementation for the writing/reading tools,
- can easily be extended.

What exists

For automata

Grail, OpenFst, FSM ...

- Have their own textual format,
- not meant to be an exchange format.

What exists

For automata

Grail, OpenFst, FSM ...

- Have their own textual format,
- not meant to be an exchange format.

For graphs

GraphML ...

- Can express complex graph structures and attach data,
- however these data are not strongly typed,
- it lacks typing conventions dedicated to automata.

FSMXML

History

- CIAA'03, exchange format idea. XML suggested (Sheng Yu).
- CIAA'04, first session on XML proposals (AutoML, FSMXML, ...).
- CIAA'05, second session on XML proposals.

FSMXML

History

- CIAA'03, exchange format idea. XML suggested (Sheng Yu).
- CIAA'04, first session on XML proposals (AutoML, FSMXML, ...).
- CIAA'05, second session on XML proposals.

Features

- Currently supporting finite weight automata, transducers and rational expressions.
- Not linked to any platform or application.
- Used and tested for communication within the Vaucanson platform.

Automaton structure


```
<automatonStruct>
  <states>
    <state id="s1"/>
    <state id="s2"/>
  </states>




</automatonStruct>
```

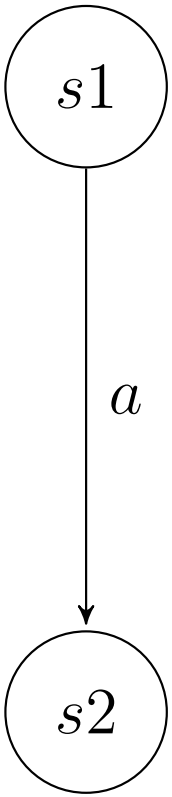
Automaton structure


```
<automatonStruct>
  <states>
    <state id="s1"/>
    <state id="s2"/>
  </states>

  <transitions>
    <transition src="s1" target="s1"/>
    <transition src="s1" target="s2"/>




  </transitions>

</automatonStruct>
```

Automaton structure

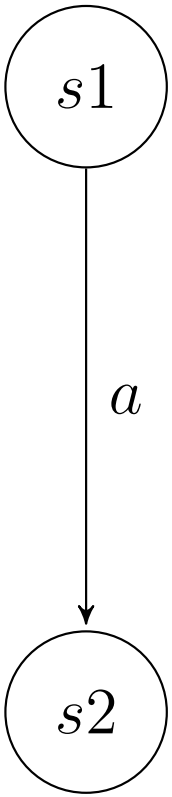

```
<automatonStruct>
  <states>
    <state id="s1"/>
    <state id="s2"/>
  </states>

  <transitions>
    <transition src="s1" target="s1"/>
    <transition src="s1" target="s2"/>

    <initial state="s1"/>
    <final state="s2"/>

  </transitions>

</automatonStruct>
```

Automaton labels : What we do not do



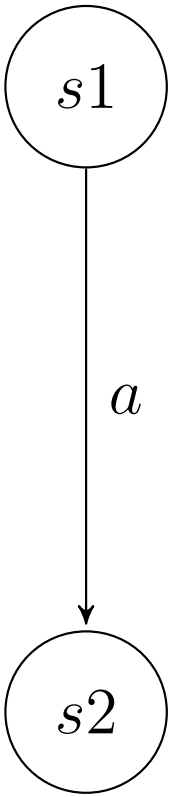

```
<transition src="s1" target="s2" label="a"/>
```

Automaton labels : What we do not do



~~<transition src="s1" target="s2" label="a"/>~~

Automaton labels : What we do not do

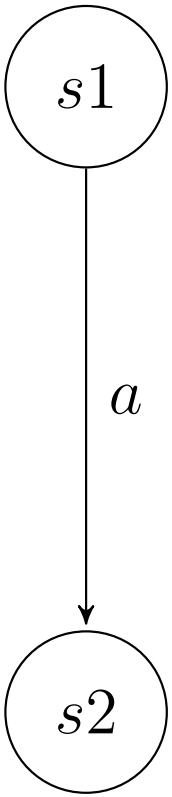



```
<transition src="s1" target="s2" label="a"/>
```

A label can be

- a letter, but also...
- a word, a pair of words (transducer), ...
- with a weight ...
- even a rational expression.

Automaton labels : What we do not do



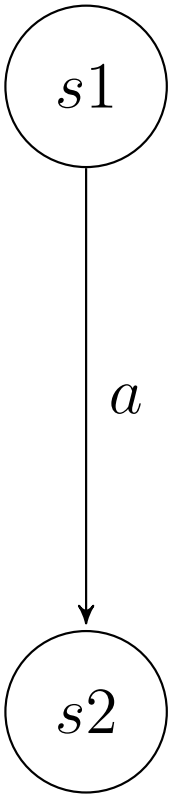

```
<transition src="s1" target="s2" label="a"/>
```

A label can be

- a letter, but also...
- a word, a pair of words (transducer), ...
- with a weight ...
- even a rational expression.

therefore we use an XML tree to describe labels.

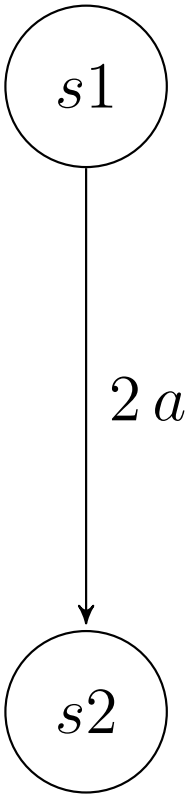
Automaton labels



A simple letter:

```
<transition src="s1" target="s2"/>
  <label>
    <monElmt>
      <monGen value="a"/>
    </monElmt>
  </label>
</transition>
```

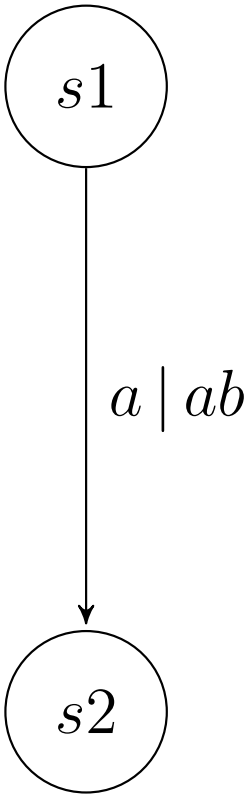
Automaton labels



A simple letter with a weight:

```
<transition src="s1" target="s2"/>
  <label>
    <leftExtMul>
      <weight value="2"/>
      <monElmt>
        <monGen value="a"/>
      </monElmt>
    </leftExtMul>
  </label>
</transition>
```

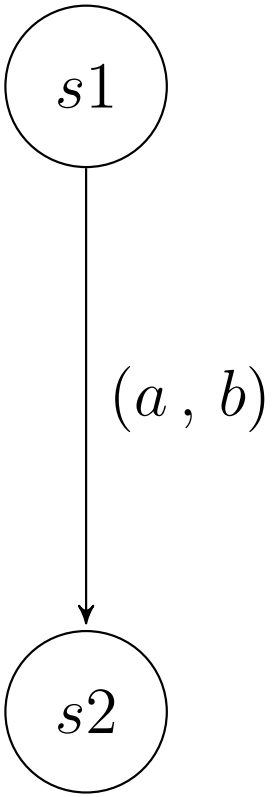
Automaton labels



An pair of words (transducer):

```
<transition src="s1" target="s2"/>
  <label>
    <monElmt>
      <monElmt>
        <monGen value="a"/>
      </monElmt>
      <monElmt>
        <monGen value="a"/>
        <monGen value="b"/>
      </monElmt>
    </monElmt>
  </label>
</transition>
```

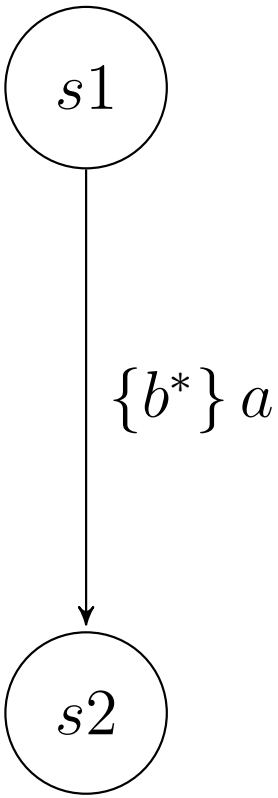
Automaton labels



Letter of complex type:

```
<transition src="s1" target="s2"/>
  <label>
    <monElmt>
      <monGen>
        <monCompGen value="a"/>
        <monCompGen value="b"/>
      </monGen>
    </monElmt>
  </label>
</transition>
```

Automaton labels



In transducers via Kleene-Schützenberger:

```
<transition src="s1" target="s2"/>
  <label>
    <leftExtMul>
      <weight>
        <star>
          <monElmt>
            <monGen value="b"/>
          </monElmt>
        </star>
      </weight>
      <monElmt>
        <monGen value="a"/>
      </monElmt>
    </leftExtMul>
  </label>
</transition>
```

Format structure for automata

```
<fsmxml xmlns="http://vaucanson.lrde.epita.fr/"
        version="1.0">

<automaton name="a1">
  <valueType>
    Automaton type
  </valueType>
  <automatonStruct>
    Automaton content
  </automatonStruct>
</automaton>

</fsmxml>
```

Format structure for regular expressions

```
<fsmxml xmlns="http://vaucanson.lrde.epita.fr/"
        version="1.0">

<regExp name="a1">
  <valueType>
    Expression type
  </valueType>
  <typedRegExp>
    Expression content
  </typedRegExp>
</regExp>

</fsmxml>
```

Automaton/Expression type

Based on free monoid

- simple letters: char, integers, digits...
- complex types of letters: tuples of simple letters,

Automaton/Expression type

Based on free monoid

- simple letters: char, integers, digits...
- complex types of letters: tuples of simple letters,

Based on product of free monoids

Automaton/Expression type

Based on free monoid

- simple letters: char, integers, digits...
- complex types of letters: tuples of simple letters,

Based on product of free monoids

With weights

- taken in numerical sets:
  $\mathbb{B}, \mathbb{N}, \mathbb{Z}, \mathbb{Q}, \ldots (\mathbb{Z}, max, +), (\mathbb{Z}, min, +), \ldots$
- regular expressions (transducers via Kleene-Schützenberger's theorem).

FSMXML

Specifications available at

`http://vaucanson.lrde.epita.fr/XML`

And is used and tested in the Vaucanson platform

`http://vaucanson.lrde.epita.fr`

FSMXML

Specifications available at

`http://vaucanson.lrde.epita.fr/XML`

And is used and tested in the Vaucanson platform

`http://vaucanson.lrde.epita.fr`

Other features

- Geometric data,
- writing direction (left, right),
- writing data (identity representation),
- ...

Further developments

Alphabet definition

- Currently supported : enumeration.
- Available but not described yet : “range”, “set”
- ”a-zA-Z” or ”alphanum”... Unicode standardized subsets ?

Further developments

Alphabet definition

- Currently supported : enumeration.
- Available but not described yet : “range”, “set”
- ”a-zA-Z” or ”alphanum”... Unicode standardized subsets ?

Drawing informations

For graphic interfaces, ...

Further developments

Alphabet definition

- Currently supported : enumeration.
- Available but not described yet : “range”, “set”
- ”a-zA-Z” or ”alphanum”... Unicode standardized subsets ?

Drawing informations

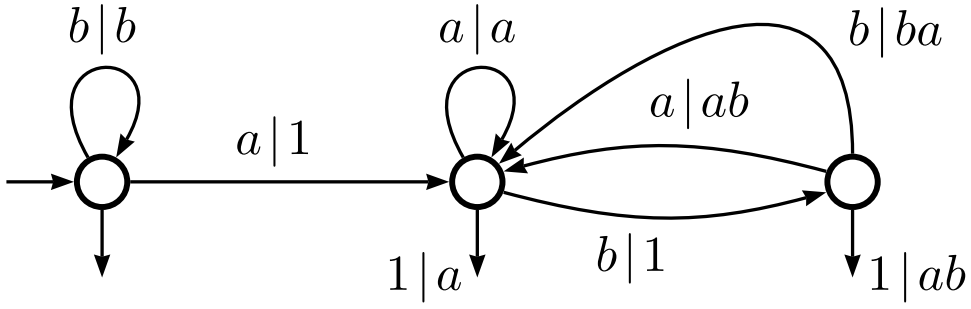
For graphic interfaces, ...

Computable properties

Deterministic, complete, normalized, trim, ...

Automaton $\mathcal{D}_2$

The transducer $\mathcal{D}_2$ that realises the Fibonacci reduction $a\,b\,b \longrightarrow b\,a\,a$, viewed as an automaton on $\{a, b\}* \times \{a, b\}*$.




```
<fsmxml  xmlns   = "http://vaucanson-project.org"
         version = "1.0" >

  <automaton name = "d2" >
    <valueType>
      <semiring  type       = 'numerical'
                 set        = 'B'
                 operations = 'classical' />
      <monoid    type       = 'product'
                 prodDim    = "2" >
        <monoid    type        = 'free'
                   genSort     = 'simple'
                   genKind     = 'letter'
                   genDescript = 'enum' >
          <monGen value="a"/>
          <monGen value="b"/>
        </monoid>
        <monoid    type        = 'free'
                   genSort     = 'simple'
                   genKind     = 'letter'
                   genDescript = 'enum' >
          <monGen value="a"/>
          <monGen value="b"/>
        </monoid>
      </monoid>
    </valueType>

    <automatonStruct>
      <states>
        <state  id="0"  name = "1" />
        <state  id="1"  name = "a" />
        <state  id="2"  name = "ab" />
      </states>
```

Automaton $\mathcal{D}_2$

```
<transitions>
  <transition  source="0"  target="0" >
    <label>
      <monElmt>
        <monElmt>
          <monGen  value = "b" />
        </monElmt>
        <monElmt>
          <monGen  value = "b" />
        </monElmt>
      </monElmt>
    </label>
  </transition>
  <transition  source="0"  target="1" >
    <label>
      <monElmt>
        <monElmt>
          <monGen  value = "a" />
        </monElmt>
        <one/>
      </monElmt>
    </label>
  </transition>
```

```
<transition  source="1"  target="1" >
  <label>
    <monElmt>
      <monElmt>
        <monGen  value = "a" />
      </monElmt>
      <monElmt>
        <monGen  value = "a" />
      </monElmt>
    </monElmt>
  </label>
</transition>
<transition  source="1"  target="2" >
  <label>
    <monElmt>
      <monElmt>
        <monGen  value = "b" />
      </monElmt>
      <one/>
    </monElmt>
  </label>
</transition>
```

Automaton $\mathcal{D}_2$

```
<transition  source="2"  target="1" >
  <label>
     <monElmt>
       <monElmt>
         <monGen  value = "a" />
       </monElmt>
       <monElmt>
         <monGen  value = "a" />
         <monGen  value = "b" />
       </monElmt>
     </monElmt>
  </label>
</transition>
<transition  source="2"  target="1" >
  <label>
     <monElmt>
       <monElmt>
         <monGen  value = "b" />
       </monElmt>
       <monElmt>
         <monGen  value = "b" />
         <monGen  value = "a" />
       </monElmt>
     </monElmt>
  </label>
</transition>
```

```
      <initial state="0"/>
      <final   state="0"/>
      <final   state="1"/>
        <label>
           <monElmt>
             <one/>
             <monElmt>
               <monGen  value = "a" />
             </monElmt>
           </monElmt>
        </label>
      </final>
      <final   state="2"/>
        <label>
           <monElmt>
             <one/>
             <monElmt>
               <monGen  value = "a" />
               <monGen  value = "b" />
             </monElmt>
           </monElmt>
        </label>
      </final>
    </transitions>
  </automatonStruct>
</automaton>

</fsmxml>
```