Finite State Machine XML (FSMXML) Specifications

The Vaucanson Group

September 8, 2008

Abstract

This document describes FSMXML. FSMXML provides generic finite state machines and regular expressions' XML description.

Contents

1 Terminology

1.1 General

- *MUST*
  This word, or the terms *REQUIRED* or *SHALL*, mean that the definition is an absolute requirement of the specification.

- *MUST NOT*
  This phrase, or the phrase *SHALL NOT*, mean that the definition is an absolute prohibition of the specification.

- *SHOULD*
  This word, or the adjective *RECOMMENDED*, mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

- *SHOULD NOT*
  This phrase, or the phrase *NOT RECOMMENDED* mean that there may exist valid reasons in particular circumstances when the particular behaviour is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behaviour described with this label.

- *MAY*
  This word, or the adjective *OPTIONAL*, mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

1.2 FSMXML

Each element of the FSMXML format is described in a section with the following 2 parts:

1.2.1 Children

Describes the required/allowed children of the element, their number of occurrences and a brief description of what they stand for.

1.2.2 Attributes

The attributes of the element are listed in a table like the following:

Name	*Info*	*Type*	*Default*	*Values*
Description				

in which:

Name

Gives the name of the attribute.

Info

Gives informations about the attribute:

- *Required*
  The attribute is mandatory.
- *Pivot*
  Depending on the attribute's value, the form of the children should differ.
- *Unique*
  When the same element can be used more than once, it means that one shall not declare twice the same value in each of those elements.
- *Valid*
  Implies that the value must have already been defined previously in the XML document.

Type

Gives the type of the value:

- *token*
  Is used for a string which takes only a couple of already defined values, listed in the *Values* column.
- *ID*
  Is used for a string which represents an identifier, that can be used many times in the document to refer to the unique same "concept".
- *generator*
  Is used for a string which only takes a couple of values (implicitly) defined in the XML document.
- *weight*
  Is used for a string which only takes a couple of values (implicitly) defined in the XML document.
- *URI*
  A Uniform Resource Identifier (URI), is a compact string of characters used to identify or name a resource.
- *integer*, *float*, *string*

Default

Since some attributes are always required and would usually take the same value, a default value is possible. Omitting the attribute will implicitly stand for the default value.

Values

When *Type* is set to `token`, lists all the possible values that can take the attribute.

Description

Gives a complete description of the attribute.

2 FSMXML

2.1 <fsmxml>

The top-level *root* element, which carries version information, etc.
See Automaton $\mathcal{A}_1$, lines 1 and 72 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
xmlns		URI	none	
Sets the namespace.				
version		float	none	
Sets the version of FSMXML used. This one is 1.0				

Children

- <regExp>, A regular expression. Occurs 0 or more times. See Section 4.1 for more details.
- <automaton>, An automaton. Occurs 0 or more times. See Section 5.1 for more details.

3 Type

Regular expressions and automata are described over a type, which can be described in the same way for both objects. In FSMXML this type is described by the following elements:

3.1 <valueType>

The top-level element of the type part.
See Automaton $\mathcal{A}_1$, lines 5-16 for a complete example.

Attributes

None.

Children

- <semiring>, The semiring. Occurs 1 time, is required. See Section 3.7 for more details.
- <monoid>, The alphabet. Occurs 1 time, is required. See Section 3.2 for more details.

3.2 <monoid>

Holds the description of a monoid.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`type`	Required, Pivot	token	none	"unit", "free", "product"
Type of the monoid.				

3.2.1 Unit monoid

When `type` is set to "unit", it is equivalent to the lack of monoids. It enables the possibility to describe valued graphs within the same format.

3.2.2 Free monoid

When `type` is set to "free", the `<monoid>` describes a free monoid and inherits the following new attributes and children.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`genKind`	Required, Pivot	token	none	"simple", "tuple"
Kind of the generators.				
`genDescript`	Required, Pivot	token	"enum"	"enum", "range", "set"
How are described the generators.				

Children

- `<writingData>`, Representation data. Occurs 0 or 1 time. See Section 6.1.1 for more details.

3.2.2.1 Free monoid with "simple" generators

When `type` is set to "free", and `genKind` to "simple", the `<monoid>` inherits the following new attributes and children.
See Automaton $\mathcal{A}_1$, lines 9-15 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`genSort`	Requied	token	none	"letter", "digit", "integer", "alphanum"
Sort of the generators.				

Children

- `<monGen>`, A monoid generator. Occurs 1 or more times. See Section 3.5 for more details.

3.2.2.2 Free monoid with "tuple" generators

When `type` is set to "free", and `genKind` to "tuple", the `<monoid>` inherits the following new attributes and children.
See Automaton $\mathcal{P}_1''$, lines 9-33 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`genDim`	Required	integer	none	Greater than 1
Dimension of the tuple.				

Children

- `<genSort>`, List of sort of generator for each "free" monoid. Occurs 1 time, is required. See Section 3.3 for more details.
- `<monGen>`, A monoid generator. Occurs 1 or more times. See Section 3.5 for more details.

3.2.3 Product Monoid

When `type` is set to "product", the `<monoid>` describes a product of free monoids and inherits the following new attributes and children.
See Automaton φ_1^{-1}, lines 9-26 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`prodDim`	Required	integer	none	Greater than 1
Dimension of the product.				

Children

- `<writingData>`, Representation data. Occurs 0 or 1 time. See Section 6.1.2 for more details.
- `<monoid>`, A `free` monoid. Occurs `prodDim` times. See Section 3.2.2 for more details.

3.3 `<genSort>`

Describes the sort of the generator of each item of the tuple in a "free" monoid with "tuple" generators.
See Automaton $\mathcal{P}_1''$, lines 13-16 for a complete example.

Attributes

None.

Children

- `<genCompSort>`, Sort of an item within the generator. Occurs `genDim` times. See Section 3.4 for more details.

3.4 `<genCompSort>`

Describes the sort of the `k`th coordinate/component in a "tuple" generator, `k` being the position of the element in the list within `<genSort>`.
See Automaton $\mathcal{P}_1''$, lines 18-19 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`value`	Required	token	none	"letter", "digit", "integer"
Sort of a coordinate/component of the "tuple" generator.				

Children

None.

3.5 `<monGen>`

3.5.1 "enum" generators

When `genDescript` is set to "enum", the `<monGen>` inherits the following new attributes:

3.5.1.1 "enum" generator of "simple" sort

Describes a generator of monoid when its `genDescript` is set to "enum" and `genKind` to "simple".
See Automaton $\mathcal{A}_1$, lines 13-14 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`value`	Required	generator	none	
Gives the value of the generator. Should fit `genSort` restriction.				

If used within a `<monoid>`, should also be *Unique*. If used within a `<monElmt>`, should also be *Valid*.

Children

None.

3.5.1.2 "enum" generator of "tuple" sort

Describes a generator of monoid when its `genDescript` is set to "enum" and `genKind` to "tuple". See Automaton φ_1^{-1}, lines 17-32 for a complete example.

Attributes

None.

Children

- `<monCompGen>`, A "tuple" monoid generator. Occurs `genDim` times. See Section 3.6 for more details.

3.5.2 "range" and "set" generators

Even if these `Tokens` are present in FSMXML, no further investigations were done. Since VAUCANSON only use enumerations, we let these investigations to more expert people.

3.6 `<monCompGen>`

Gives the `k`th coordinate/component in a "tuple" generator, `k` being the position of the element in the list within `<monGen>`.
See Automaton φ_1^{-1}, lines 18-19 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`value`	Required	generator	none	
Gives the value of the coordinate/component. Should fit the associated `genCompSort`.				

Children

None.

3.7 `<semiring>`

Holds a semiring description.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`type`	Required, Pivot	token	none	"numerical", "series"
Type of the semiring.				

Children

- `<writingData>`, Representation data. Occurs 0 or 1 time. See Section 6.1.2 for more details.

3.7.1 Numerical semiring

When `type` is set to "numerical", `<semiring>` describes a numerical semiring and inherits the following new attributes and children.
See Automaton $\mathcal{A}_1$, lines 6-8 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`set`	Required	token	none	"B", "N", "Z", "Q", "R", "C"
The set on which is described the semiring.				
`operation`	Required	token, string	none	"classical", "minPlus", "maxPlus"
Set the operations to work with into the semiring. This is not an exhaustive list.				

Children

None.

3.7.2 Series semiring

When `type` is set to "series", the `<semiring>` describes a series semiring and inherits the following new attributes and children.
See Automaton $\mathcal{D}_2$ (bis), lines 6-19 for a complete example.

Attributes

None.

Children

- `<semiring>`, A semiring. Occurs 1 time. See Section 3.7 for more details.
- `<monoid>`, A monoid. Occurs 1 time. See Section 3.2 for more details.

4 Regular Expressions

4.1 `<regExp>`

Holds the complete representation of a regular expression.
See A rational expression for $|\mathcal{C}_1| = (a + b)^*(b \cdot (2\,a + 2\,b)^*)$, lines 4-52 for a complete example.

Attributes

None.

Children

- `<valueType>`, The regular expression's type. Occurs 1 time. See Section 3.1 for more details.
- `<typedRegExp>`, The regular expression's body. Occurs 1 time. See Section 4.2 for more details.

4.2 `<typedRegExp>`

Holds the typed regular expression.
See A rational expression for $|\mathcal{C}_1| = (a + b)^*(b \cdot (2\,a + 2\,b)^*)$, lines 17-51 for a complete example.

Attributes

None.

Children

- Typed regular expression. Occurs 1 time. See Section 4.3 for more details.

4.3 Regular expression's body

A regular expression's body is represented by a recursive tree of elements, listed below. Any of these elements is seen as a "Typed regular expression".
See A rational expression for $|\mathcal{C}_1| = (a + b)^*(b \cdot (2\,a + 2\,b)^*)$, lines 18-50 for a complete example.

4.3.1 `<sum>`

Sum of two expressions.

Attributes

None.

Children

- Left Typed regular expression. Occurs 1 time. See Section 4.3 for more details.
- Right Typed regular expression. Occurs 1 time. See Section 4.3 for more details.

4.3.2 `<product>`

Product of two expressions.

Attributes

None.

Children

- Left Typed regular expression. Occurs 1 time. See Section 4.3 for more details.
- Right Typed regular expression. Occurs 1 time. See Section 4.3 for more details.

4.3.3 `<star>`

Star of an expression.

Attributes

None.

Children

- Typed regular expression to starify. Occurs 1 time. See Section 4.3 for more details.

4.3.4 `<rightExtMul>` and `<leftExtMul>`

Represents the right/left scalar multiplication of an expression.

Attributes

None.

Children

- `<weight>`, Weight for the multiplication. Occurs 1 time. See Section 4.4 for more details.
- Typed regular expression to multiply. Occurs 1 time. See Section 4.3 for more details.

4.3.5 `<monElmt>`

Represents a monoid element, which is a concatenation of monoid generators.

4.3.5.1 On a free monoid

Represents a monoid element on a free monoid.
See Automaton $\mathcal{A}_1$, lines 26-28 for a complete example.

Attributes

None.

Children

- `<monGen>`, A monoid Generator. Occurs as many time as wanted. See Section 3.5 for more details.

4.3.5.2 On a product monoid.

Represents a monoid element on a product monoid.
See Automaton φ_1^{-1}, lines 35-42 for a complete example.

Attributes

None.

Children

- `<monElmt>` or `<one>` A monoid element or the identity of a `free` monoid. Occurs `prodDim` times. See paragraph 4.3.5.1 and Section 4.3.7 for more details.

4.3.6 `<zero>`

Represents the null series.

Attributes

None.

Children

None.

4.3.7 `<one>`

Represents the identity series or the identity symbol of a `free` monoid.

Attributes

None.

Children

None.

4.4 `<weight>`

Represents the weight of an expression.

4.4.1 On a numerical semiring

Represents the weight of an expression with a numerical semiring.
See Automaton $\mathcal{C}_1$, lines 47-47 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`value`	Required	weight	none	
Weight's value.				

Children

None.

4.4.2 On a series semiring

Represents the weight of an expression with a series semiring.
See Automaton $\mathcal{D}_2$ (bis), lines 39-43 for a complete example.

Attributes

None.

Children

- Typed regular expression taken into the semiring. Occurs 1 time. See Section 4.3 for more details.

5 Automata

5.1 `<automaton>`

Holds the complete representation of an automaton.
See Automaton $\mathcal{A}_1$, lines 4-70 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`name`		string		
Name of the automaton.				
`readingDir`		token	left	"left", "right"
Reading direction of the automaton.				

Children

- `<geometricData>`, The automaton's geometry data. Occurs 0 or 1 time. See Section 6.2.1 for more details.
- `<drawingData>`, The automaton's drawing data. Occurs 0 or 1 time. See Section 6.3 for more details.
- `<valueType>`, The automaton's type. Occurs 1 time. See Section 3.1 for more details.
- `<automatonStruct>`, The automaton's content. Occurs 1 time. See Section 5.2 for more details.

5.2 `<automatonStruct>`

Holds the automaton's content.
See Automaton $\mathcal{A}_1$, lines 17-69 for a complete example.

Attributes

None.

Children

- `<states>`, Lists the automaton's states. Occurs 1 time. See Section 5.3 for more details.
- `<transitions>`, Lists the automaton's transitions. Occurs 1 time. See Section 5.5 for more details.

5.3 `<states>`

Holds the automaton's states.
See Automaton $\mathcal{A}_1$, lines 18-22 for a complete example.

Attributes

None.

Children

- `<state>`, Adds a state. Occurs 0 or more times. See Section 5.4 for more details.

5.4 `<state>`

Describes a state. `key` is usually used to define in which order should be processed the states. `name` allows you to give a more explicit name.
See Automaton $\mathcal{A}_1$, lines 19-21 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`id`	Required, Unique	ID	none	
Id of the state.				
`key`		integer	0	Positive integer
Possible key of the state.				
`name`		string	none	
Name of the state.				

Children

- `<geometricData>`, Adds geometry data. Occurs 0 or 1 time. See Section 6.2.2 for more details.
- `<drawingData>`, Adds drawing data. Occurs 0 or 1 time. See Section 6.3 for more details.

5.5 `<transitions>`

Holds the automaton's transitions and initial/final properties of states.
See Automaton $\mathcal{A}_1$, lines 23-68 for a complete example.

Attributes

None.

Children

- `<transition>`, Adds a transition. Occurs 0 or more times. See Section 5.6 for more details.
- `<initial>`, Adds the initial property to a state. Occurs 0 or more times. See Section 5.7 for more details.
- `<final>`, Adds the final property to a state. Occurs 0 or more times. See Section 5.7 for more details.

5.6 `<transition>`

Describes a transition.
See Automaton $\mathcal{A}_1$, lines 24-30 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`source`	Required	ID	none	Valid ID
Source of the transition.				
`target`	Required	ID	none	Valid ID
Target of the transition.				

Children

- `<label>`, Label of the transition. Occurs 1 time See Section 5.8 for more details.
- `<geometricData>`, Adds geometry data. Occurs 0 or 1 time. See Section 6.2.3 for more details.
- `<drawingData>`, Adds drawing data. Occurs 0 or 1 time. See Section 6.3 for more details.

5.7 `<initial>` and `<final>`

Adds the initial/final property to a state.
See Automaton $\mathcal{A}_1$, lines 66-67 for a complete example.

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`state`	Required	ID	none	Valid ID
State to give the initial/final property.				

Children

- `<label>`, Label of the transition. Occurs 1 time See Section 5.8 for more details.
- `<geometricData>`, Adds geometry data. Occurs 0 or 1 time. See Section 6.2 for more details.
- `<drawingData>`, Adds drawing data. Occurs 0 or 1 time. See Section 6.3 for more details.

5.8 `<label>`

Holds the label of a transition.
See Automaton $\mathcal{A}_1$, lines 25-29 for a complete example.

Attributes

None.

Children

- Typed regular expression. Occurs 1 time. See Section 4.3 for more details.

6 Automaton's optional properties

6.1 `<writingData>`

Stores informations that might be useful for input / output.

6.1.1 for monoid

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`identitySymbol`		string		
How to display the monoid identity.				

Children

None.

6.1.2 for semiring

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`identitySymbol`		string		
How to display the semiring identity.				
`zeroSymbol`		string		
How to display the semiring zero.				

Children

None.

6.2 <geometricData>

Stores informations on geometric representation.

6.2.1 for the automaton

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
x		float		
Gives the relative horizontal origin.				
y		float		
Gives the relative vertical origin.				

Children

None.

6.2.2 for states

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
x		float		
Gives state horizontal position.				
y		float		
Gives state vertical position.				

Children

None.

6.2.3 for the transitions

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
TransitionType	Pivot	token	none	“EdgeL”, “EdgeR”, “ArcL”, “ArcR”, “Loop”
The type of the transition.				

`loopDir`		integer		Between 0 and 360
Only available if “Loop” is set. The angle of the loop direction.				
`labelPos`		float		Between 0 and 100
Position of the label on the arrow. 0 is the source, 100 the target.				
`labelDist`		float		Greater than 0
Distance between the arrow and the label.				

Children

None.

6.2.4 for initial states

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`initialDir`		integer		Between 0 and 360
The angle of the input arrow.				
`labelPos`		float		Between 0 and 100
Position of the label on the arrow. 0 is the source, 100 the target.				
`labelDist`		float		Greater than 0
Distance between the arrow and the label.				

Children

None.

6.2.5 for final states

Attributes

Name	*Info*	*Type*	*Default*	*Values*
Description				
`finalDir`		integer		Between 0 and 360
The angle of the output arrow.				
`finalMod`		token		“circle”,“arrow”
How should be represented a final state.				
`labelPos`		float		Between 0 and 100
Position of the label on the arrow. 0 is the source, 100 the target.				
`labelDist`		float		Greater than 0
Distance between the arrow and the label.				

Children

None.

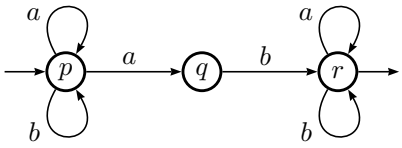
6.3 <drawingData>

Not yet specified.

A Examples

A.1 Automaton $\mathcal{A}_1$

Ref: ETA p. 58 Fig. I.1.1

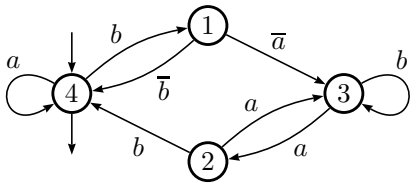



```
<fsmxml  xmlns   = "http://vaucanson-project.org"
         version = "1.0" >

<automaton name = "a1" >
  <valueType>
    <semiring  type        = 'numerical'
               set         = 'B'
               operations  = 'classical' />
    <monoid    type        = 'free'
               genSort     = 'simple'
               genKind     = 'letter'
               genDescript = 'enum' >
      <monGen value="a"/>
      <monGen value="b"/>
    </monoid>
  </valueType>
```

```
    <automatonStruct>
      <states>
        <state  id="0"  name="p"/>
        <state  id="1"  name="q"/>
        <state  id="2"  name="r"/>
      </states>
      <transitions>
        <transition  source="0"  target="0" >
          <label>
            <monElmt>
              <monGen value="a"/>
            </monElmt>
          </label>
        </transition>
        <transition  source="0"  target="0" >
          <label>
            <monElmt>
              <monGen value="b"/>
            </monElmt>
          </label>
        </transition>
        <transition  source="0"  target="1" >
          <label>
            <monElmt>
              <monGen value="a"/>
            </monElmt>
          </label>
        </transition>
        <transition  source="1"  target="2" >
          <label>
            <monElmt>
              <monGen value="b"/>
            </monElmt>
          </label>
        </transition>
        <transition  source="2"  target="2" >
          <label>
            <monElmt>
              <monGen value="a"/>
            </monElmt>
          </label>
        </transition>
        <transition  source="2"  target="2" >
          <label>
            <monElmt>
              <monGen value="b"/>
            </monElmt>
          </label>
        </transition>
        <initial  state="0"/>
        <final    state="2"/>
      </transitions>
    </automatonStruct>
  </automaton>

  </fsmxml>
```

A.2 Automaton $\mathcal{P}_1''$

Ref: HECCA p. 14 Fig. 7(a)

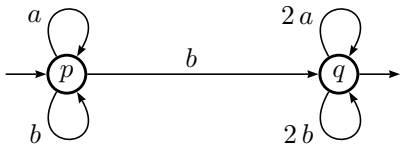



```
<fsmxml  xmlns   = "http://vaucanson-project.org"
         version = "1.0" >

<automaton name = "Pdp1" >
  <valueType>
    <semiring  type       = 'numerical'
               set        = 'B'
               operations = 'classical' />
    <monoid    type       = 'free'
               genSort    = 'tuple'
               genDescript = 'enum'
               genDim     = "2" >
      <genSort>
         <genCompSort value = 'letter' />
         <genCompSort value = 'digit'  />
      </genSort>
      <monGen>
         <monCompGen value = "a" />
         <monCompGen value = "0" />
      </monGen>
      <monGen>
         <monCompGen value = "a" />
         <monCompGen value = "1" />
      </monGen>
      <monGen>
         <monCompGen value = "b" />
         <monCompGen value = "0" />
      </monGen>
      <monGen>
         <monCompGen value = "b" />
         <monCompGen value = "1" />
      </monGen>
    </monoid>
  </valueType>
```

```
    <automatonStruct>
      <states>
      <state  id="0"  name="1"/>
      <state  id="1"  name="2"/>
      <state  id="2"  name="3"/>
      <state  id="3"  name="4"/>
      </states>
      <transitions>
        <transition  source="3"  target="3" >
          <label>
            <monElmt>
              <monGen>
                <monCompGen value = "a" />
                <monCompGen value = "0" />
              </monGen>
            </monElmt>
          </label>
        </transition>
        <transition  source="3"  target="0" >
          <label>
            <monElmt>
              <monGen>
                <monCompGen value = "b" />
                <monCompGen value = "0" />
              </monGen>
            </monElmt>
          </label>
        </transition>
        <transition  source="0"  target="3" >
          <label>
            <monElmt>
              <monGen>
                <monCompGen value = "b" />
                <monCompGen value = "1" />
              </monGen>
            </monElmt>
          </label>
        </transition>
        <transition  source="0"  target="2" >
          <label>
            <monElmt>
              <monGen>
                <monCompGen value = "a" />
                <monCompGen value = "1" />
              </monGen>
            </monElmt>
          </label>
        </transition>
        ...
        <initial state="3"/>
        <final   state="3"/>
      </transitions>
    </automatonStruct>
  </automaton>

</fsmxml>
```

A.3 Automaton $\mathcal{C}_1$

Ref: ETA p. 437 Fig. III.2.6




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

<automaton name = "c1" >
  <valueType>
    <semiring  type       = 'numerical'
               set        = 'N'
               operations = 'classical' />
    <monoid    type       = 'free'
               genSort    = 'simple'
               genKind    = 'digit'
               genDescript = 'enum' >
      <monGen value="0"/>
      <monGen value="1"/>
    </monoid>
  </valueType>
```

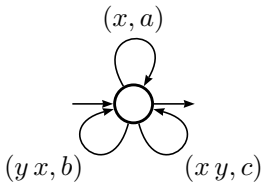
```
    <automatonStruct>
      <states>
        <state  id="0"  name="p"/>
        <state  id="1"  name="q"/>
      </states>
      <transitions>
        <transition  source="0"  target="0" >
          <label>
            <monElmt>
              <monGen value="0"/>
            </monElmt>
          </label>
        </transition>
        <transition  source="0"  target="0" >
          <label>
            <monElmt>
              <monGen value="1"/>
            </monElmt>
          </label>
        </transition>
        <transition  source="0"  target="1" >
          <label>
            <monElmt>
              <monGen value="1"/>
            </monElmt>
          </label>
        </transition>
        <transition  source="1"  target="1" >
          <label>
            <leftExtMul>
              <weight value="2"/>
              <monElmt>
                <monGen value="0"/>
              </monElmt>
            </leftExtMul>
          </label>
        </transition>
        <transition  source="1"  target="1" >
          <label>
            <leftExtMul>
              <weight value="2"/>
              <monElmt>
                <monGen value="1"/>
              </monElmt>
            </leftExtMul>
          </label>
        </transition>
        <initial  state="0"/>
        <final    state="1"/>
      </transitions>
    </automatonStruct>
  </automaton>

</fsmxml>
```

A.4 A rational expression for $|\mathcal{C}_1| = (a+b)^*(b\cdot(2\,a+2\,b)^*)$

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

<regExp name = "c1-behaviour" >
  <valueType>
    <semiring  type        = 'numerical'
               set         = 'N'
               operations  = 'classical' />
    <monoid    type        = 'free'
               genSort     = 'simple'
               genKind     = 'digit'
               genDescript = 'enum' >
      <monGen value="0"/>
      <monGen value="1"/>
    </monoid>
  </valueType>
  <typedRegExp>
    <product>
      <star>
        <sum>
          <monElmt>
            <monGen value="0"/>
          </monElmt>
          <monElmt>
            <monGen value="1"/>
          </monElmt>
        </sum>
      </star>
      <product>
        <monElmt>
          <monGen value="1"/>
        </monElmt>
        <star>
          <sum>
            <leftExtMul>
              <weight value="2"/>
              <monElmt>
                <monGen value="0"/>
              </monElmt>
            </leftExtMul>
            <leftExtMul>
              <weight value="2"/>
              <monElmt>
                <monGen value="1"/>
              </monElmt>
            </leftExtMul>
          </sum>
        </star>
      </product>
    </product>
  </typedRegExp>
</regExp>

</fsmxml>
```

A.5 Automaton φ_1^{-1}

Ref: ETA p. 582 (not quite) The one state automaton that realises the inverse of the morphism φ_1.

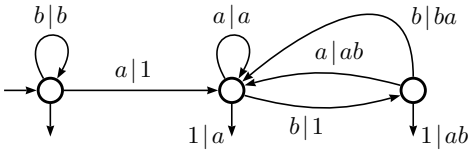



```
<fsmxml  xmlns   = "http://vaucanson-project.org"
         version = "1.0" >

<automaton name = "phim1" >
  <valueType>
    <semiring  type       = 'numerical'
               set        = 'B'
               operations = 'classical' />
    <monoid    type       = 'product'
               prodDim    = "2" >
      <monoid    type        = 'free'
                 genSort     = 'simple'
                 genKind     = 'letter'
                 genDescript = 'enum' >
        <monGen value="x"/>
        <monGen value="y"/>
      </monoid>
      <monoid    type       = 'free'
                 genSort    = 'simple'
                 genKind    = 'letter'
                 genDescrip = 'enum' >
        <monGen value="a"/>
        <monGen value="b"/>
        <monGen value="c"/>
      </monoid>
    </monoid>
  </valueType>
```

```
    <automatonStruct>
      <states>
        <state  id="0" />
      </states>
      <transitions>
        <transition  source="0"  target="0" >
          <label>
            <monElmt>
              <monElmt>
                <monGen  value = "x" />
              </monElmt>
              <monElmt>
                <monGen  value = "a" />
              </monElmt>
            </monElmt>
          </label>
        </transition>
        <transition  source="0"  target="0" >
          <label>
            <monElmt>
              <monElmt>
                <monGen  value = "y" />
                <monGen  value = "x" />
              </monElmt>
              <monElmt>
                <monGen  value = "b" />
              </monElmt>
            </monElmt>
          </label>
        </transition>
        <transition  source="0"  target="0" >
          <label>
            <monElmt>
              <monElmt>
                <monGen  value = "x" />
                <monGen  value = "y" />
              </monElmt>
              <monElmt>
                <monGen  value = "c" />
              </monElmt>
            </monElmt>
          </label>
        </transition>
        <initial state="0"/>
        <final   state="0"/>
      </transitions>
    </automatonStruct>
  </automaton>

  </fsmxml>
```

A.6 Automaton $\mathcal{D}_2$

Ref: ETA p. 699 Fig. V.1.4
The transducer $\mathcal{D}_2$ that realises the Fibonacci reduction $a\,b\,b \to b\,a\,a$, viewed as an automaton on $\{a,b\}* \times \{a,b\}*$.



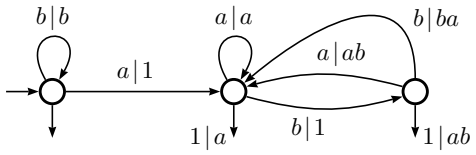

```
<fsmxml  xmlns   = "http://vaucanson-project.org"
         version = "1.0" >

<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>
```

```
<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>
```

```
        <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>
```

A.7 Automaton $\mathcal{D}_2$ (bis)

Ref: ETA p. 699 Fig. V.1.4
The transducer $\mathcal{D}_2$ that realises the Fibonacci reduction $a\,b\,b \to b\,a\,a$, viewed as an automaton on $\{a, b\}*$ with multiplicities in Rat $\{a, b\}*$.




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

<automaton name = "d2-rw" >
  <valueType>
    <semiring  type       = 'series'
               zeroSymbol     = "0"
               identitySymbol = "1" >
      <semiring  type       = 'numerical'
                 set        = 'B'
                 operations = 'classical' />
      <monoid    type       = 'free'
                 genSort    = 'simple'
                 genKind    = 'letter'
                 genDescript = 'enum' >
        <monGen value="a"/>
        <monGen value="b"/>
      </monoid>
    </semiring>
    <monoid    type       = 'free'
               genSort    = 'simple'
               genKind    = 'letter'
               genDescript = 'enum'>
      <monGen value="a"/>
      <monGen value="b"/>
    </monoid>
  </valueType>

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

```
<transitions>
  <transition  source="0"  target="0" >
    <label>
      <leftExtMul>
        <weight>
          <monElmt>
            <monGen  value = "b" />
          </monElmt>
        </weight>
        <monElmt>
          <monGen  value = "b" />
        </monElmt>
      </leftExtMul>
    </label>
  </transition>
  <transition  source="0"  target="1" >
    <label>
      <monElmt>
        <monGen  value = "a" />
      </monElmt>
    </label>
  </transition>
  <transition  source="1"  target="1" >
    <label>
      <leftExtMul>
        <weight>
          <monElmt>
            <monGen  value = "a" />
          </monElmt>
        </weight>
        <monElmt>
          <monGen  value = "a" />
        </monElmt>
      </leftExtMul>
    </label>
  </transition>
  <transition  source="1"  target="2" >
    <label>
      <monElmt>
        <monGen  value = "b" />
      </monElmt>
    </label>
  </transition>
```

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

  </fsmxml>
```