Proposal: an XML representation for automata

The Vaucanson Group <vaucanson@lrde.epita.fr>

CIAA, July 24, 2004

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

Introduction

CIAA 2003:

- *Introducing Vaucanson*,
- Need for the definition of an XML formalism for automata.

This proposal:

- Based on the experience capitalized on VAUCANSON,
- Partially implemented in VAUCANSON,

Motivations

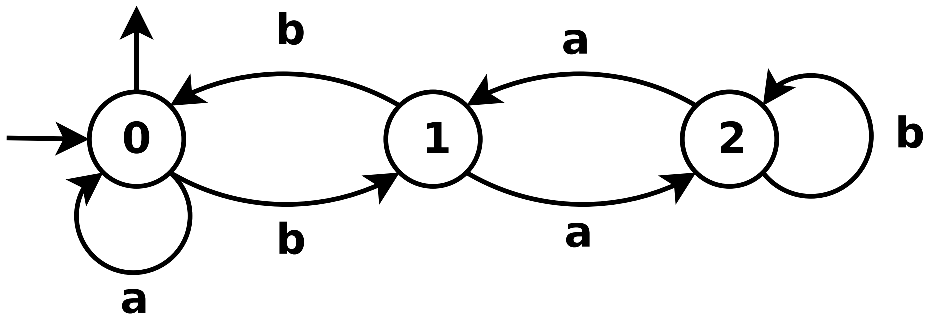
- Inter-program communication,
- unified format for all automata,
- keep persistent data between different executions of a program.

Why XML?

- Well known format,
- easy to parse (many libraries),
- easy to make a DTD.

Table of Contents

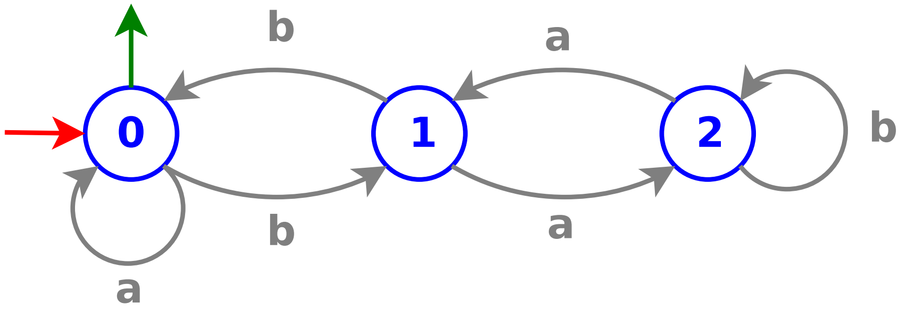
Overview



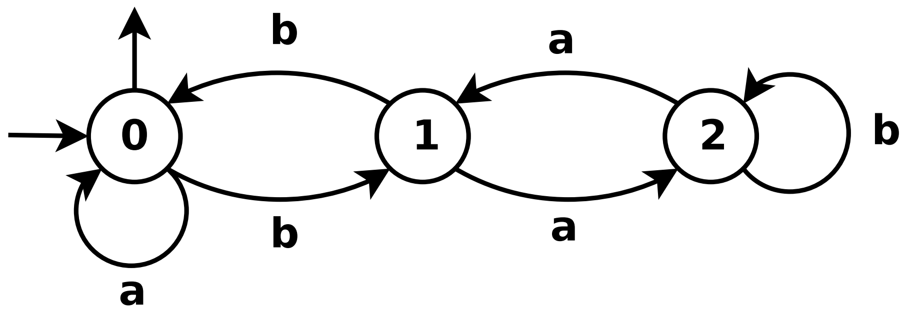

```
<automaton>

  <type> ... </type>

  <content> ... </content>

</automaton>
```

Content

```
<automaton>

  <type> ... </type>

  <content>

    <states> ... </states>
    <transitions> ... </transitions>
    <initials> ... </initials>
    <finals> ... </finals>

  </content>

</automaton>
```

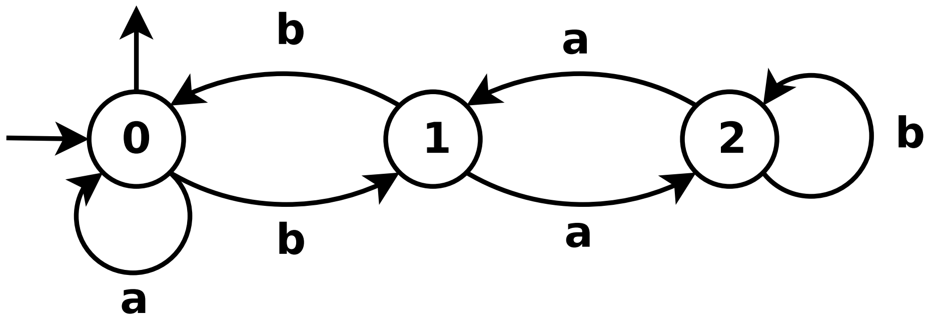


States and transitions



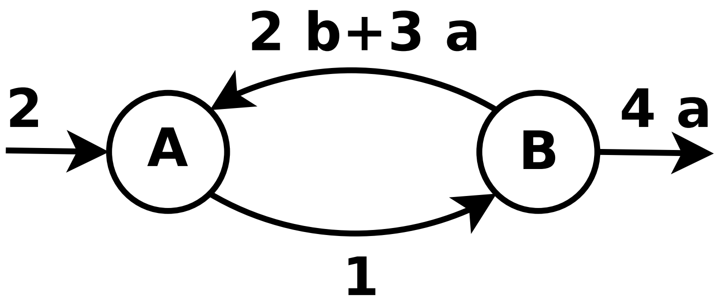

```
<states>
  <state name="0" />
  <state name="1" />
  <state name="2" />
</states>

<transitions>
  <transition src="0" label="a" dst="0" />
  <transition src="0" label="b" dst="1" />
  <transition src="1" label="a" dst="2" />
  <transition src="1" label="b" dst="0" />
  <transition src="2" label="a" dst="1" />
  <transition src="2" label="b" dst="2" />
</transitions>
```

Initials and finals



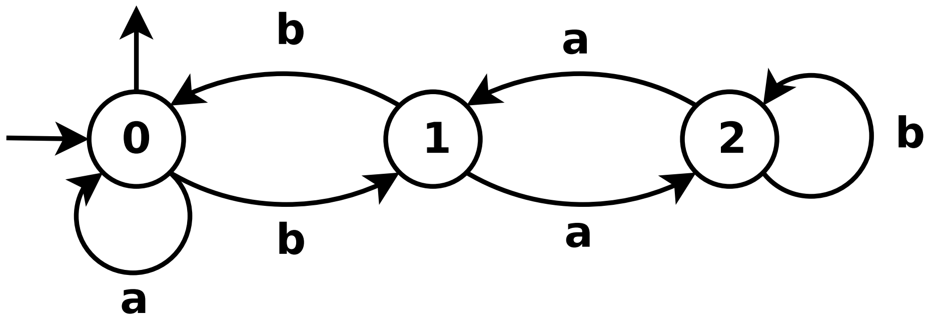

```xml
<initials>
  <initial state="0"/>
</initials>

<finals>
  <final state="0"/>
</finals>
```

Advanced content



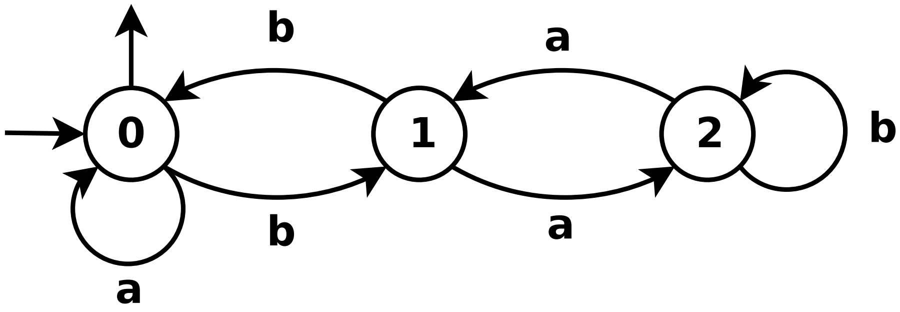

```
<states>
  <state name="A" />
  <state name="B" />
</states>

<transitions>
  <transition src="A" dst="B" />
  <transition src="B" label="2 b+3 a" dst="A" />
</transitions>

<initials><initial label="2" state="A"/></initials>

<finals><final label="4 a" state="B"/></finals>
```

Overview



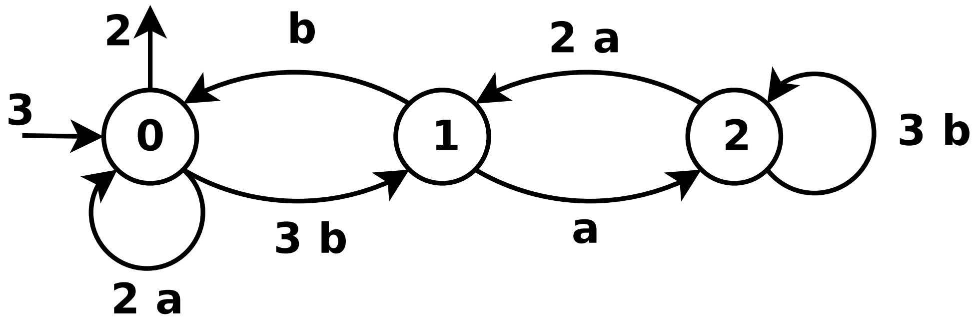

```
<automaton>

  <type> ... </type>

  <content> ... </content>

</automaton>
```

Type




```
<type>
  <monoid>
    <generator value="a"/>
    <generator value="b"/>
  </monoid>
</type>
```

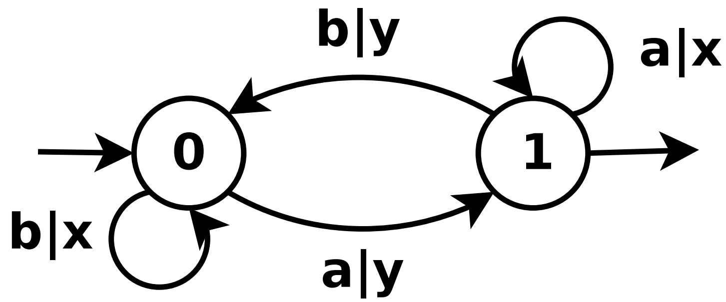
Type: weighted automata




```
<type>
  <monoid>
    <generator value="a" />
    <generator value="b" />
  </monoid>
  <semiring set="Z" operations="numerical" />
</type>
```

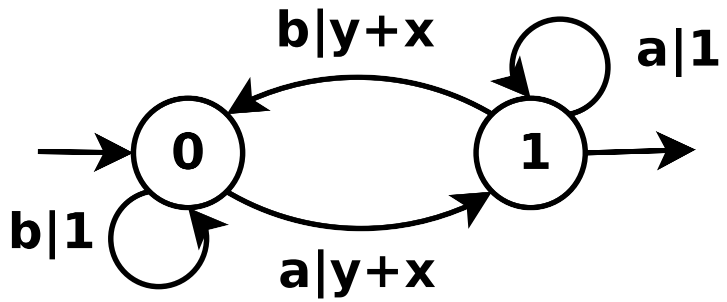
Type: transducers

```
<monoid type="product">

  <monoid>
    <generator value="a" />
    <generator value="b" />
  </monoid>

  <monoid>
    <generator value="x" />
    <generator value="y" />
  </monoid>

</monoid>
```



Type: transducers as automata over boolean series

```
<type>
  <monoid>
    <generator value="a" />
    <generator value="b" />
  </monoid>

  <semiring set="ratseries">
    <monoid>
      <generator value="x" />
      <generator value="y" />
    </monoid>
    <semiring set="B" operations="boolean" />
  </semiring>
</type>
```

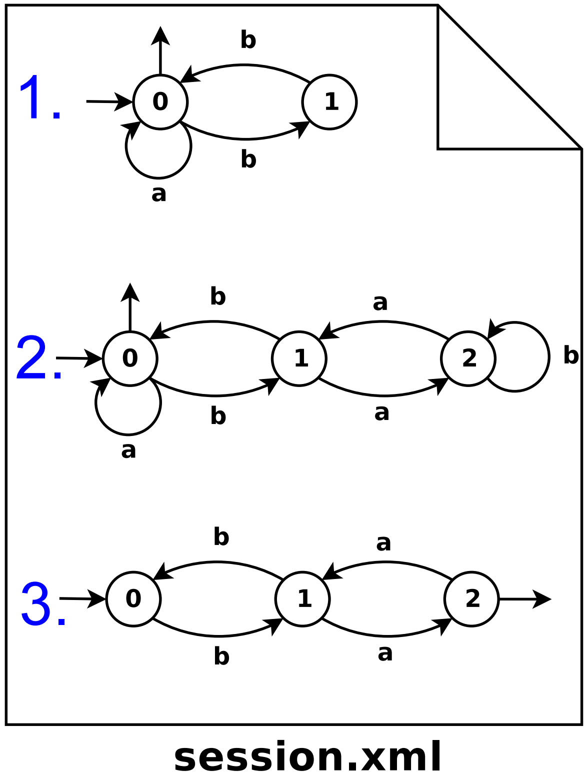


Extra features: geometry

```
<automaton><geometry StateFillColor="blue" />
  ...
  <states>
    <state name="s0"><geometry x="10" y="10" /></state>
    ...
  </states>

  <transitions><geometry EdgeLineStyle="dashed" />
    ...
  </transitions>
  ...
</automaton>
```

Extra features: sessions

```
<session>
  <automaton>
    ...
  </automaton>

  <automaton>
    ...
  </automaton>

  <automaton>
    ...
  </automaton>
</session>
```



Conclusion

- The formalism is simple
- it is consistent and powerful
- partially supported in the VAUCANSON generic library,
- needs to be extended to support more kinds of automata.