Table des matières

Annexe A

User's guide

A.1 Installation

A.1.1 Download instructions

The archive can be downloaded from the ssh server of the ENST. Since the project is still under development, and not yet included in Vaucanson development process, you need an ENST account to retrieve it.

```
$> scp login@ssh.enst.fr:/tmp/vgi-dev.tar.gz .
```

In a close future, the archive will be available from the LRDE web page of Vaucanson project (http ://vaucanson.lrde.epita.fr)

A.1.2 Needed softwares

In order to compile the project, you need the following tools :
- Java SDK 1.4 or higher (including java, javac and jar),
- a C++ compiler supported by Vaucanson (gcc 3.2 or higher, for example),
- Vaucanson 0.6 or higher and all related softwares.

A.1.3 Build instructions

As any project using the Autotools, the build process is approximatively the same.

```
$> tar xzf vgi-dev.tar.gz
$> ./configure
$> make
```

If you want to use Vaucanson algorithms, and use a special version of Vaucanson, you should try :

```
$> ./configure --with-vcsn=<your vaucanson include directory>
```

If you have any problem during the configuration, please see README and INSTALL files. Notice that GNU make is preferred to other make.

A.1.4 Load and save an automaton

To load an automaton, click on File →Open...

Then choose a valid XML file in the file chooser window.

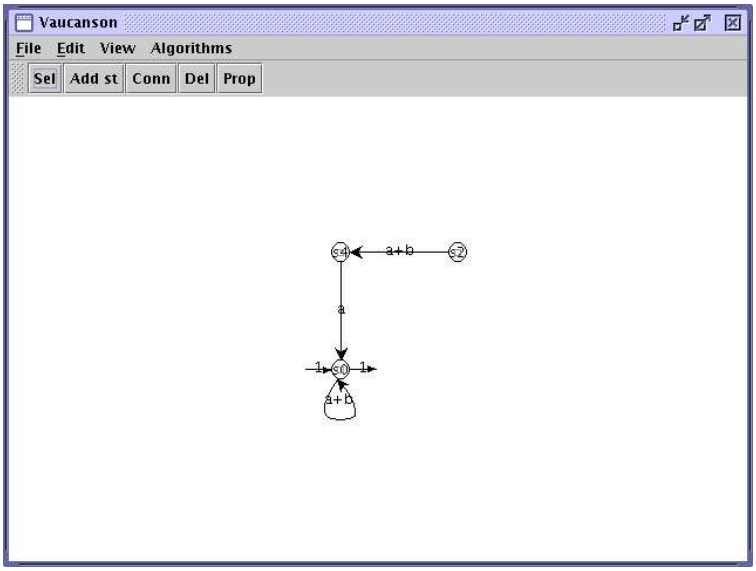

FIG. A.1 – A sample automaton

To save an automaton, just click on File → Save as...

A.1.5 Launch an algorithm from Vaucanson

At the moment, only few algorithms are available. You can find them in the Algorithms menu. To launch an algorithm, just choose it in the menu. If the algorithm expects arguments, a popup is shown to retrieve them.

Algorithms available :

– Determinization,
– Minimization,
– Trim.

These algorithms will be completed with many more very soon.

A.1.6 Launch an user-defined algorithm

First, you have to ensure that the external program fits the following requirements :
- read a Vaucanson-XML file on standard input,
- write a Vaucanson-XML file on standard output,
- read arguments from the command line.

Then, three steps must be accomplished :
- set the program to launch with Algorithms →User defined... →Choose program...,
- if needed, set the command line arguments with Algorithms →User defined... →Set command line arguments...,
- launch the algorithm with Algorithms →User defined... →Launch algorithm.

A.2 Layout management

One of the features of the interface is to provide layout algorithm, to embed your XML file with geometric coordinates for automaton. To do so, you have to select which states are to be moved, and launch the approriate algorithm in the following list :
- circle layout,
- line layout,
- quad layout,
- global layout.

For circle layout, you are prompted for the radius of the circle. You can leave to automatic.

For quad layout, you are prompted for the number of lines you wish to have. You can leave to automatic.

For global layout, the layout apply to the whole automaton. This is still in development.

For example :

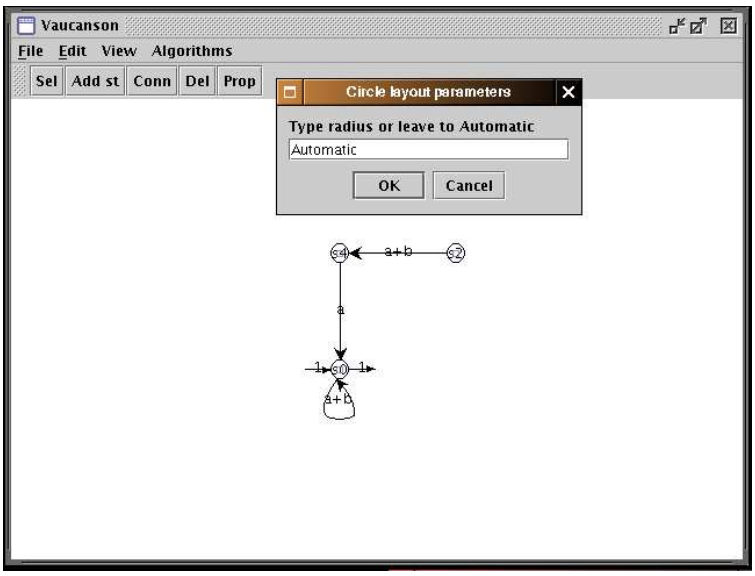

FIG. A.2 – Apply circle layout

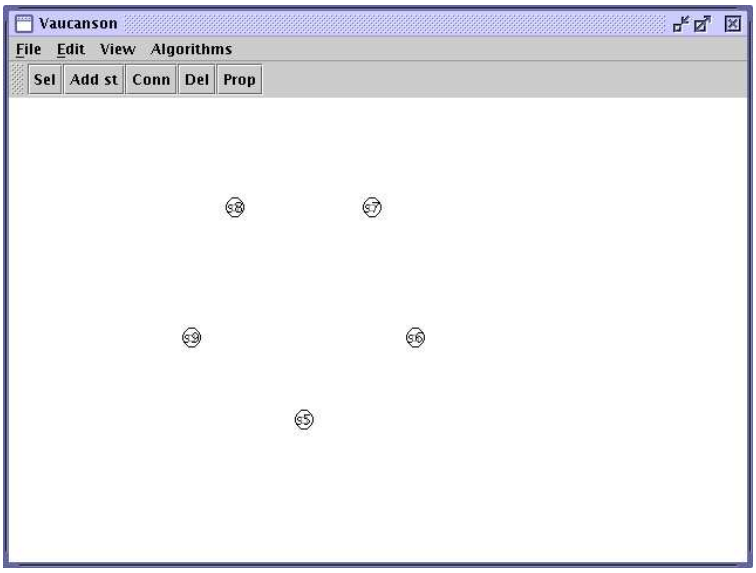

FIG. A.3 – Circle layout result

A.3 Your first automaton

First, create an empty automaton with File →New

A.3.1 Adding states

To add state, you can either

– click on Edit →Create states

– clic on Add St on the toolbar.

If you choose to do it through the menu, you will be prompted to set which disposition and how many states you want.

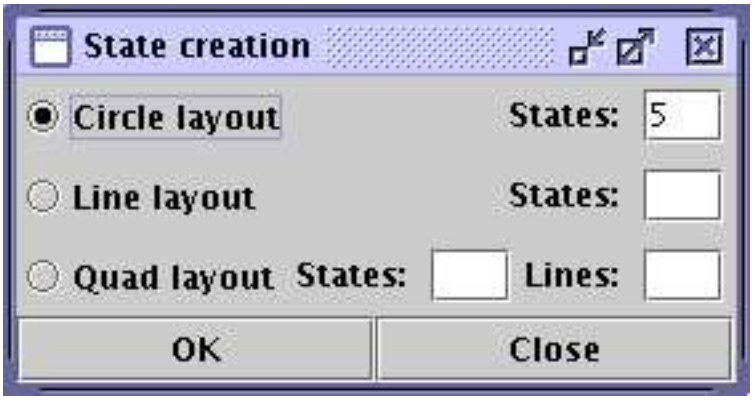

FIG. A.4 – Create states

But if you choose to go with the toolbar button, notice that :

– you enter create state mode, meaning that when you click anywhere in the drawing area, a state is created,

– you have to click on the button Sel on the toolbar to exit the create state mode.

To edit state properties, for example to set a state as initial, you have to select it with the mouse, and then do one of the following :

– click on Prop button in the toolbar,

or

– click on Properties in the contextual menu (right click),

or

– click on Edit →Properties in the menu.

A window is shown :

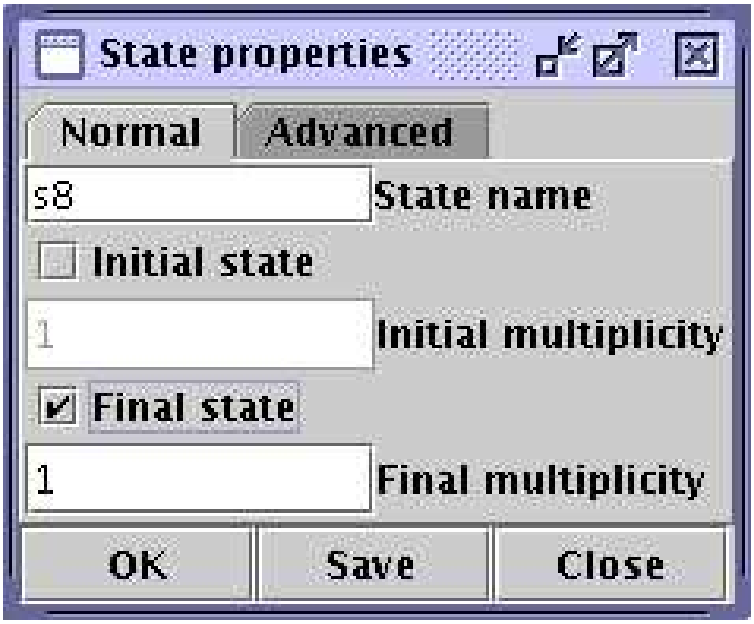

FIG. A.5 – State properties

A.3.2 Connecting states

A.3.3 Connecting two states

You have to select exactly two states, and do one of the following :

– click on Conn button in the toolbar,

or

– click on Connect in the contextual menu (right click),

or

– click on Edit →Connect in the menu.

To edit transition properties, you have to do the same thing than for state properties edition.

A.3.4 Creating loop

You have to select exactly one state, and do one of the following :

– click on Conn button in the toolbar,

or

– click on Connect in the contextual menu (right click),

or

– click on Edit →Connect in the menu.

If you want to edit loop properties, you have also special properties like look size or loop direction. These informations can be carried in the XML format.

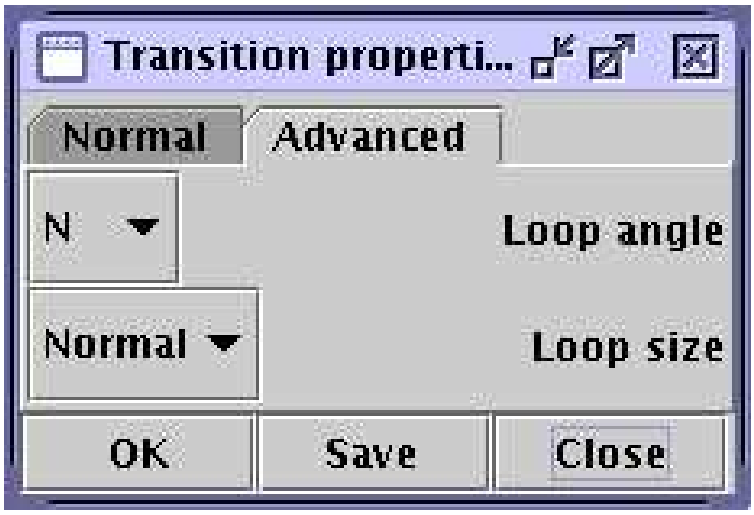

FIG. A.6 – Loop properties

A.3.5 Deleting elements

To delete any element, you just have to select them and do one of the following :

– click on Del button in the toolbar,

or

– click on Delete in the contextual menu (right click),

or

– click on Edit →Delete in the menu.

A.3.6 Automaton algebraic structure

One of the most powerful feature of the Vaucanson-XML document model is the support of algebraic type informations of the automaton that are forced to be with its content. The different possibilities are fixed in the DTD.

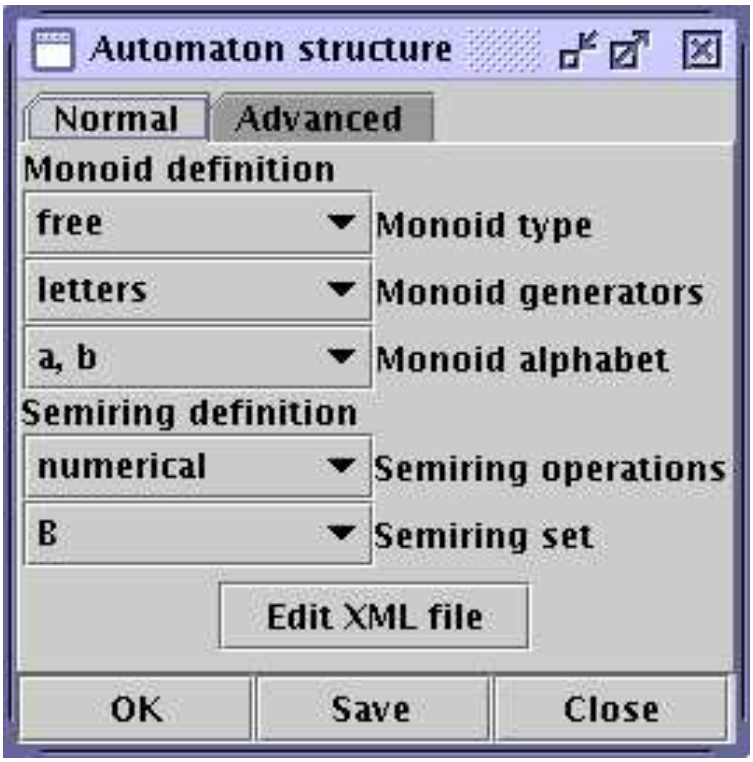

FIG. A.7 – Automaton structure

A.3.7 In-place editing

The interface is written in order to support in-place editing. To edit transition label, or state name, you can double-click on an element. An in-place dialog is opened.

A.4 Informations and bug report

To report a bug, please mail at vaucanson-bug@lrde.epita.fr

To get more information, please mail the developer at loulou@lrde.epita.fr

Annexe B

Vaucanson DTD

```
<!--
  This DTD describes Vaucanson automaton. The aim of this representation is
  to be able to describe any automaton

  -->

<!-- Document structure -->
<!ELEMENT session (geometry?, automaton*)>
<!ELEMENT automaton (geometry?, type, content)>
<!ELEMENT type (monoid, semiring?)>
<!ELEMENT content (geometry?, states, transitions, initials, finals)>
<!ELEMENT monoid (generator*|monoid*)>
<!ELEMENT semiring (monoid?, semiring?)>
<!ELEMENT states (geometry?, state*)>
<!ELEMENT transitions (geometry?, transition*)>
<!ELEMENT state (geometry?)>
<!ELEMENT transition (geometry?)>
<!ELEMENT initials (geometry?, initial*)>
<!ELEMENT finals (geometry?, final*)>
<!ELEMENT generator EMPTY>
<!ELEMENT initial (geometry?)>
<!ELEMENT final (geometry?)>
<!ELEMENT geometry EMPTY>

<!ATTLIST geometry
StateLabelColor CDATA #IMPLIED
StateLabelScale CDATA #IMPLIED
StateLineStyle CDATA #IMPLIED
StateLineWidth CDATA #IMPLIED
StateLineColor CDATA #IMPLIED
StateFillStatus CDATA #IMPLIED
StateFillColor CDATA #IMPLIED
```

```
DimStateLineStyle CDATA #IMPLIED
DimStateLineColor CDATA #IMPLIED
DimStateLineCoef CDATA #IMPLIED
DimStateLabelColor CDATA #IMPLIED
DimStateFillColor CDATA #IMPLIED
StateLineDblCoef CDATA #IMPLIED
StateLineDblSep CDATA #IMPLIED
EdgeLabelColor CDATA #IMPLIED
EdgeLabelScale CDATA #IMPLIED
EdgeLineStyle CDATA #IMPLIED
EdgeLineWidth CDATA #IMPLIED
EdgeLineColor CDATA #IMPLIED
ArcAngle CDATA #IMPLIED
LArcAngle CDATA #IMPLIED
ArcCurvature CDATA #IMPLIED
EdgeOffset CDATA #IMPLIED
ArcOffset CDATA #IMPLIED
LoopOffset CDATA #IMPLIED
ForthBackEdgeOffset CDATA #IMPLIED
DimEdgeLineStyle CDATA #IMPLIED
DimEdgeLineCoef CDATA #IMPLIED
DimEdgeLineColor CDATA #IMPLIED
DimEdgeLabelColor CDATA #IMPLIED
EdgeBorderCoef CDATA #IMPLIED
EdgeBorderColor CDATA #IMPLIED
EdgeLineDoubleCoefOne CDATA #IMPLIED
EdgeLineDoubleCoefTwo CDATA #IMPLIED
ZZSize CDATA #IMPLIED
ZZShape CDATA #IMPLIED
ZZLineWidth CDATA #IMPLIED
TransLabelZZCoef CDATA #IMPLIED
LargeScale CDATA #IMPLIED
MediumScale CDATA #IMPLIED
SmallScale CDATA #IMPLIED
TinyScale CDATA #IMPLIED
MediumStateDiameter CDATA #IMPLIED
SmallStateDiameter CDATA #IMPLIED
LargeStateDiameter CDATA #IMPLIED
VerySmallStateDiameter CDATA #IMPLIED
VSStateLineCoef CDATA #IMPLIED
ArrowOnMediumState CDATA #IMPLIED
ArrowOnSmallState CDATA #IMPLIED
ArrowOnLargeState CDATA #IMPLIED
ArrowOnVerySmallState CDATA #IMPLIED
LoopOnMediumState CDATA #IMPLIED
LoopOnSmallState CDATA #IMPLIED
```

```
LoopOnLargeState CDATA #IMPLIED
LoopOnVariableState CDATA #IMPLIED
CLoopOnMediumState CDATA #IMPLIED
CLoopOnSmallState CDATA #IMPLIED
CLoopOnLargeState CDATA #IMPLIED
CLoopOnVariableState CDATA #IMPLIED
EdgeLabelPosit CDATA #IMPLIED
EdgeLabelRevPosit CDATA #IMPLIED
ArcLabelPosit CDATA #IMPLIED
ArcLabelRevPosit CDATA #IMPLIED
LArcLabelPosit CDATA #IMPLIED
LArcLabelRevPosit CDATA #IMPLIED
LoopLabelPosit CDATA #IMPLIED
LoopLabelRevPosit CDATA #IMPLIED
CLoopLabelPosit CDATA #IMPLIED
CLoopLabelRevPosit CDATA #IMPLIED
InitStateLabelPosit CDATA #IMPLIED
InitStateLabelRevPosit CDATA #IMPLIED
FinalStateLabelPosit CDATA #IMPLIED
FinalStateLabelRevPosit CDATA #IMPLIED
EdgeArrowWidth CDATA #IMPLIED
EdgeArrowLengthCoef CDATA #IMPLIED
EdgeDblArrowWidth CDATA #IMPLIED
EdgeDblArrowLengthCoef CDATA #IMPLIED
EdgeArrowInsetCoef CDATA #IMPLIED
EdgeArrowStyle CDATA #IMPLIED
EdgeRevArrowStyle CDATA #IMPLIED
StateDimen CDATA #IMPLIED
StateDblDimen CDATA #IMPLIED
left CDATA #IMPLIED
bottom CDATA #IMPLIED
top CDATA #IMPLIED
right CDATA #IMPLIED
label CDATA #IMPLIED
x CDATA #IMPLIED
y CDATA #IMPLIED
curvature CDATA #IMPLIED
curvside CDATA #IMPLIED
direction CDATA #IMPLIED
doubleline CDATA #IMPLIED
>

<!ATTLIST session
        xmlns   CDATA   #FIXED "http://www.lrde.epita.fr/vaucanson"
>
<!-- attributes of the root node -->
```

```
<!--
  xmlns:
    this attribute is used to embed automaton into another document
  type:
    Here are some properties of that can determinize the possible
    implementation.
  labels:
    Pure means here "without any spontaneous transition"
  -->
<!ATTLIST automaton
        xmlns   CDATA   #FIXED "http://www.lrde.epita.fr/vaucanson"

        name    ID      #IMPLIED
>

<!--
Monoid node.
  -->
<!ATTLIST monoid
        type    (unit|free|product|CPFM|FCM|FC)
                        "free"
        generators
                (letters|pairs|weigthed|integers)
                        "letters"
>

<!--
Semiring node.
If you want to write a tranducer, you have to specify that the semiring is
constructed from a free monoid.
  -->
<!ATTLIST semiring
        operations
        (boolean|numerical|tropicalMax|tropicalMin|function|hadamard|shuffle)
                        "numerical"
        set     (B|Z|R|ratseries)
                        "B"
>

<!ATTLIST generator
        value   CDATA   #REQUIRED
>

<!--
        Weight is used only when labels are couples. (For series and polynoms,
        please use regular expression in labels instead).
```

```
  -->
<!ENTITY % serie-value
        'label   CDATA   "1"
         weight  CDATA   #IMPLIED'
>

<!--
        Please number all or no state.
        In relative mode, positionment is made like in a table.
  -->
<!ATTLIST state
        name    ID      #REQUIRED
        number  CDATA   #IMPLIED
        label   CDATA   #IMPLIED
>

<!ATTLIST transition
        name    ID      #IMPLIED
        src     IDREF   #REQUIRED
        dst     IDREF   #REQUIRED

        %serie-value;
>

<!ATTLIST initial
        state   IDREF   #REQUIRED
        %serie-value;
>

<!ATTLIST final
        state   IDREF   #REQUIRED
        %serie-value;
>
```