VAUCANSON-G
A package for drawing automata and graphs
User’s manual

Sylvain Lombardy       Jacques Sakarovitch
IGM-LabInfo, Univ. Paris-Est    LTCI, CNRS / TELECOM ParisTech

October 27., 2008

Version 0.4

— o —

VAUCANSON-G is a package that allows to draw automata within texts written using \LaTeX. It is a set of macros that uses commands of the beautiful PSTricks package, due to Timothy van Zandt\textsuperscript{1} and which is part of the current \LaTeX distribution. VAUCANSON-G is the result of the experience gained in composing the several hundreds of figures in the book *Eléments de théorie des automates* ([5]) and in several other papers and conferences slides of the authors.

The design of VAUCANSON-G implements the following underlying philosophy:

1. ‘simple’ automata should be described with simple commands.

2. Every element of a figure can easily be given a style (which differs from the implicit one).

3. The complexity of commands (or the number of things to be remembered to use them) should gradually grow with the complexity of the figure composed by these commands.

4. It should be possible to handle the figures both in size and appearance without modifying them.

The following example\textsuperscript{2} shows how a simple automaton can be drawn with commands, in which only the minimal information needed (position and label of states, shape and label of transitions) is made explicit.

\textsuperscript{1}The maintenance and evolution of PSTricks has now been taken over by H. Voss.

\textsuperscript{2}somewhat artificial but which exhibits most of the simple commands.
The objectives of V\textsc{auCanS\textsc{on-G}} are achieved by the implicit definition of a large number of parameters that control the geometry of the figure: size of states, width of lines, \textit{etc.} and by the definition of commands that allow to handle and modify these parameters.

The following pages gives a presentation of V\textsc{auCanS\textsc{on-G}} that gradually introduces the commands from the basic ones that allow to draw already a good deal of automata to the more sophisticated ones that are used to modify the internal parameters.

\textbf{Contents}

0 Downloading and using the package 3

1 Basic commands 4

2 Using preset parameters 7

3 Composing a picture 15

4 Parametrized drawing commands 18

5 Modifying and setting the parameters 22

6 Using “plain” \textsc{PStricks} commands 30

A Version history 31

B Style files 32
0 Downloading and using the package

The \texttt{Vaucanson-G} package is downloadable from the URL:

\begin{verbatim}
/igm.univ-mlv.fr/~lombardy/Vaucanson-G/
\end{verbatim}

and consists, beside the present manual, in seven files: the wrapper \texttt{vaucanson-g.sty}, the main macro file \texttt{VauCanSon-G.tex}, and five other ancillary files\footnote{Cf. Section 5 for an explanation on the role of these files.} for options, patches and colors: \texttt{VCPref-default.tex}, \texttt{VCPref-slides.tex}, \texttt{VCPref-beamer.tex}, \texttt{VCPref-mystyle.tex}, \texttt{VCColor-names.def} that should all be stored in the user’s \texttt{texinputs} directory.

A \LaTeX{} file that will use \texttt{Vaucanson-G} macros should contain the call to the wrapper in the preamble:

\begin{verbatim}
\usepackage{vaucanson-g}
\end{verbatim}

The \texttt{Vaucanson-G} commands are to be used in a \LaTeX{} file and not in a plain \TeX{} file for it uses the macro \verb|\newcommand| that does not exist in \TeX{}.

\texttt{Vaucanson-G} is based on the \texttt{PSTricks} package. The knowledge of this macro package, described in \cite{1, 3, 4, 6}, is not necessary to use the \texttt{Vaucanson-G} commands.\footnote{It probably helps in understanding how they are built, and is useful if one wants to make figures that consists of automata but also contains other graphical elements (\textit{cf.} Section 6).} On the contrary, \texttt{Vaucanson-G} has somehow be written to save on it. In particular, the \texttt{Vaucanson-G} commands follow the simple (and rigid) syntax of \LaTeX{} commands concerning the way the arguments are specified, and not the much more flexible syntax of \texttt{PSTricks} itself.

But the call to \texttt{PSTricks} commands has the consequence that \texttt{Vaucanson-G} commands generate pieces of \texttt{Postscript} files which in turn has two main outcomes\footnote{which a user should have in mind before sending complaint and request.}:

\begin{itemize}
  \item \texttt{Vaucanson-G} is not compatible with \texttt{pdflatex};
  \item the figures produced with \texttt{Vaucanson-G} are not likely to be seen on a \texttt{DVI} viewer: the space occupied by the figure will appear empty and, in most cases, the labels will appear at the bottom of it.
\end{itemize}

\texttt{Vaucanson-G} can be used with the \texttt{beamer} document class to design slides. The document class should be called with \texttt{xcolor=pst,dvips} options:

\begin{verbatim}
\documentclass[xcolor=pst,dvips]{beamer}
\end{verbatim}

Thus equipped, we may begin.

\footnotesize

\begin{verbatim}
\usepackage{vaucanson-g}
\end{verbatim}

\normalsize
1 Basic commands

An automaton is a labelled directed graph where some of the vertices, called states, are distinguished to be initial or final states. The basic commands allow then to define and to draw states, to give them the quality of being initial or final and to give them labels as well. They allow to draw transitions between states (that have been defined before); there are 3 kinds of transitions: (straight) edges, arcs and loops.

These commands have to be put in an environment that defines a picture. All graphical parameters of these elements (size, line width, etc.) are kept implicit when using these basic commands.

1.1 The VCPicture environment

\begin{VCPicture}{(x_0, y_0)(x_1, y_1)}
\end{VCPicture}

Defines a box that “encloses” the elements defined in the environment. Its lower left (resp. upper right) corner has coordinates \((x_0, y_0)\) (resp. \((x_1, y_1)\)).

The \(x_i\) and \(y_i\) are either length or numbers in which case they represent the corresponding length in centimeter (which is the implicit length unit used in \textsc{VauCanSon-G}). The length of the box is thus \(x_1 - x_0\), its height \(y_1 - y_0\), its baseline point stands at the middle of the height. As usual in \TeX, a picture may well “contain” an element that actually lays outside of the box.

Note that the environment has only one argument consisting of two point coordinates enclosed in adjacent parentheses.

1.2 The State commands

States are represented by circles, inside which an information (the Label) can be written. Every state is given a Name which is used then internally in the figure to describe the transitions that start from it or end at it.

\texttt{State\[Label\]}{(x, y)}{(Name)}

Defines a state with name Name and draws it at the coordinate \((x, y)\) with label Label written inside.

The parameter Label is optional; default value is the empty string. It is written in ‘\TeX mathmode’.

\texttt{Initial\[Dir\]}{(Name)}

Makes the state Name initial, \textit{i.e.} draws an incoming arrow from the direction Dir to state Name.

\texttt{Final\[Dir\]}{(Name)}

Makes the state Name final, \textit{i.e.} draws an outgoing arrow from state Name in the direction Dir.
The parameter Dir is optional; the possible values are the eight main compass points: n, s, e, w, ne, nw, se and sw. Default value is w for Initial, e for Final.

Many authors, in particular those dealing with automata on infinite words, rather indicate that a state is final by drawing it with doubleline. The reason why it is not possible to achieve this result by a sequence of two commands comes from the PSTricks macros that are called by the \texttt{\LaTeX} commands.

\begin{verbatim}
\begin{VCPicture}{(0,-2)(6,2)}
% states
\State[p]{(0,1)}{A} \State{(2,-1)}{B} \State[r]{(4,1)}{C}
\FinalState[r]{(6,-1)}{D}
% initial--final
\InitialL{A} \Final{A}
\Initial[ne]{B} \Initial[nw]{B}
\Initial[se]{B} \Initial[sw]{B}
\Initial{C} \Final[w]{C} \Initial[n]{D} \Final[s]{D}
%\end{VCPicture}
\end{verbatim}

1.3 The Edge commands

Edges, like the other transitions, are oriented from the origin state to the end state. The left and the right sides of a transition are defined with respect to this orientation, being the left and the right hand of someone standing on the origin state and following the transition towards the end state.

\begin{verbatim}
\EdgeL[Pos]{Start}{End}{Label}
\EdgeR[Pos]{Start}{End}{Label}
\end{verbatim}

Draws a straight transition from the state \texttt{\textit{Start}} to the state \texttt{\textit{End}} with the label \texttt{\textit{Label}} on the left side for \texttt{\edgeL}, on the right side for \texttt{\edgeR}.

The parameter Pos is optional. It is a number \((0 \leq Pos \leq 1)\) that defines the position of the label on the edge. Default value is \(.45\) and may be used most of the time — this slight disymmetry gives some “dynamic” to the figure. Explicit value for Pos is used to avoid collision with other elements of the figure. As for states, the label is written in \texttt{\LaTeX} mathmode.
1.4 The Arc commands

Arcs are transitions that link the Start state to the End state with a curved line. The main parameter of such a curve is the angle made at both ends by the curve with the straight line that goes from Start to End and it can be oriented to the left or to the right. In the sequel, the letter $X$ stands for either L or R.

\begin{VCPicture}{(0,-2)(6,2)}
% states
\State{A}{{(0,0)}} \State{B}{{(2,-2)}} \State{C}{{(4,2)}} \State{D}{{(6,0)}}
% transitions
\ArcR{A}{B}{a} \ArcL{A}{C}{b} \LArcR{B}{D}{c} \LArcL{C}{D}{d}
\ArcL{A}{B}{a'} \EdgeL{A}{C}{b'} \EdgeR{B}{D}{c'} \LArcR{C}{D}{d'}
\end{VCPicture}

\begin{enumerate}
\item Same syntax as \texttt{\EdgeX}.
\item The starting angle is 15° for \texttt{\ArcX}, 30° for \texttt{\LArcX}.
\item Parameter Pos is optional. Default value is .4.
\end{enumerate}

1.5 The Loop commands

Loops are transitions that link a state to itself, a state which is thus both the origin and the end of the transition. The two main parameters of such a loop are the opening angle and the direction of the loop. In the sequel, the letter $Y$ stands for any of the compass points: N, S, E, W, NE, NW, SE or SW.

\begin{VCPicture}{(0,-2)(6,2)}
% states
\State{A}{{(0,0)}} \State{B}{{(2,-2)}} \State{C}{{(4,2)}} \State{D}{{(6,0)}}
% initial-final
\Initial{A} \Final[s]{{B}} \Final[n]{{C}}
% transitions
\ArcR{A}{{B}}{a} \ArcL{A}{{C}}{b} \LArcR{B}{{D}}{c} \LArcL{C}{{D}}{d}
\ArcL{A}{{B}}{a'} \EdgeR{A}{{C}}{b'} \EdgeL{B}{{D}}{c'} \LArcR{C}{{D}}{d'}
\end{VCPicture}

\begin{enumerate}
\item Draw a loop around the state Name with label Label.
\item The direction of the loop is given by $Y$. The opening of the loop is 60° for the \texttt{\LoopY} commands, 44° for the \texttt{\CLoopY} commands.
\item Parameter Pos is optional. Default value is .25.
\end{enumerate}
1.6 An example

This is an automaton with multiplicity in \( \mathbb{N} \). The multiplicity of a word \( u \) is the square of the number represented by \( u \) in base 2, when \( a \) is interpreted as 0 and \( b \) as 1.

2 Using preset parameters

Together with the basic commands, \texttt{VauCanSonG} gives easy access to two convenient parameters for drawing automata: global scaling and state size.

2.1 Global scaling

The meticulous reader has noticed that the actual size, measured for instance by the distance between the states, is not the one given in the source code. This is because we have slightly cheated and the source code of the picture is indeed encapsulated in a \texttt{VCDraw} command that uses a preset parameter \texttt{VCScale} as shown below, with no cheating anymore.
Frames that appear on figures below are an option of \texttt{V\textsc{auCanS\textsc{on}-G}} that we shall see later. They point out the effect of the scale.

There is no reason indeed for an author to “program” the drawing of an automaton at the size it will appear on his, or her, paper. On the contrary, he, or she, is likely to sketch the figure on a ruled paper at a larger scale, giving the points to be defined integer coordinates. And then she, or he, will try to put the figure within the text at a reasonable size and thus would like to scale it.

The parameter \texttt{\textbackslash VCScale} contains this global scaling factor and is initialized to the value 0.6, which corresponds to the command \texttt{\textbackslash MediumPicture}. Three other commands: \texttt{\textbackslash TinyPicture}, \texttt{\textbackslash SmallPicture} and \texttt{\textbackslash LargePicture} are available, that set \texttt{\textbackslash VCScale} to the values 0.42, 0.5 and 0.85 respectively.

These command have to be used outside the scope of the \texttt{VCPicture} environment; and this environment should be called as the parameter of a \texttt{VCDraw}.

In the sequel, and unless otherwise stated, all the figure will be given at the \texttt{\textbackslash MediumPicture} scale without mentioning the command \texttt{\textbackslash VCDraw} in the source code.

### 2.2 State size

There are three preset sizes for the states, that are called by the commands \texttt{\textbackslash SmallState}, \texttt{\textbackslash MediumState} and \texttt{\textbackslash LargeState}, that call for states of diameter 0.6cm, 0.9cm and 1.2cm respectively. The implicit size, \textit{i.e.} the one that states have if no explicit command is given, is \texttt{\textbackslash MediumState}. The “small” states are not likely to receive labels, unless their size are changed by other commands.
When called outside the scope of the environment `\VCPicture` the commands `\SmallState`, `\MediumState` and `\LargeState` set the implicit size of the states in all pictures from that point on.

\begin{VCPicture}{(-1,-1)(1,1)}\State[(0,0)]{A}\end{VCPicture}

\begin{VCPicture}{(-1,-1)(1,1)}\State[(0,0)]{A}\end{VCPicture}

\begin{VCPicture}{(-1,-1)(1,1)}\State[(0,0)]{A}\end{VCPicture}

2.3 Variable size states

One can draw states which will be called variable states and whose width is variable and fits the width of the label.\footnote{This makes sense for long labels and it may be the case then that the label is not a mathematical variable or expression any more but simple text. In order to write labels in \TeX\ LR mode, the easiest way is to write it as the parameter of a `\text` command, provided the `amsmath` package has been called in the preamble.} The height of these variable states is fixed by the current preset state size.

\texttt{StateVar[Label]{(x,y)}{Name}} \quad \text{Same as syntax as `State`}.

Note that the current version of \textsc{VauCanSon-G} uses a trick to make these states with adaptable width. They look like Egyptian cartouches but they are indeed rectangles which can be observed when transitions reach these states in an oblique way.

Arcs between variable states of different width will look asymmetric and often ungainly. Other loops than “north” and “south” loops should not be drawn around variable states. Moreover, in order to draw loops that have the same appearance as loops around standard states, one should use `\LoopVarN` or `\LoopVarS` that have the same syntax as `\LoopN` or `\LoopS`. Another possibility is to call the `\VarLoopOn` command to make all the subsequent loops suited to variable states. Switching back to ‘normal’ loops is made by `\VarLoopOff`.

For the same reason, it is \textit{not possible} to make a variable state initial or final with diagonal arrows (\textsc{ne}, \textsc{nw}, \textsc{se}, \textsc{sw}).
There exists also a `\FinalStateVar` command, with the same syntax as `\StateVar` and which draws a variable state with double line, (independently of the current drawing style for the state), exactly in the same way as does `\FinalState`.

### 2.4 Very small states

There is actually a fourth size for states, called *very small* states. To define such a state one uses the command `\VSState` with the same syntax as `\State`. However, it is not possible, of course, to put a label inside such a state.

### 2.5 Miscellaneous

A few commands allow to tune the drawing of an automaton without going to precise mastering of the parameters involved in the commands and that we shall describe in section 4.

**Dimmed states and transitions**

`\DimState` \ `\DimEdge` \ `\RstState` \ `\RstEdge` Draw states and transitions and their respective labels in gray. Restore the normal style.

May be used for instance in order to indicate the non accessible part of an automaton.

\(^7\)Since version 0.3 only.
\begin{VCPicture}{(0,-2)(6,2)}
% states
\State[A]{(0,0)}{A} \State[B]{(2,-2)}{B} \State[C]{(4,2)}{C} \\
\DimState \State[D]{(6,0)}{D}
\% initial-final
\Initial{A} \Final[s]{B} \Final[n]{C}
% transitions
\ArcR{A}{B}{a} \ArcL{A}{C}{b} \\
\DimEdge \LArcR{B}{D}{c} \LArcL{C}{D}{d} \RstEdge \\
\ArcL{A}{B}{a'} \EdgeR{A}{C}{b'} \\
\DimEdge \EdgeL{B}{D}{c'} \LArcR{C}{D}{d'}
\%
\end{VCPicture}

\begin{VCPicture}{(0,-2)(6,2)}
% states
\State[A]{(0,0)}{A} \StateLineDouble \State[B]{(2,-2)}{B} \StateLineSimple \State[C]{(4,2)}{C} \\
\StateLineDouble \State[D]{(6,0)}{D}
% initial-final
\Initial{A} \Final[s]{B}
% transitions
\ArcR{A}{B}{a} \ArcR{A}{D}{a'} \ArcL{B}{D}{c} \ArcL{C}{D}{d} \\
\DimEdge \EdgeR{A}{D}{c'} \LArcR{B}{C}{b'} \EdgeL{B}{D}{c'} \LArcR{C}{D}{d'}
\%
\end{VCPicture}

\textbf{Double lines}
\begin{itemize}
  \item \StateLineDouble \StateLineSimple \StateLineDouble \StateLineSimple
\end{itemize}

Draw states and transitions with double lines. Draw states and transitions with simple lines.

May be used for emphasize transitions or states of an automaton.
\begin{VCPicture}{(0,-2)(6,2)}
% states
\State[A]{(0,0)}{A} \StateLineDouble \State[B]{(2,-2)}{B} \StateLineSimple \State[C]{(4,2)}{C} \\
\StateLineDouble \State[D]{(6,0)}{D}
\% initial-final
\Initial{A} \Final{s}{B}
% transitions
\ArcR{A}{B}{a} \ArcR{A}{D}{a'} \ArcL{B}{D}{c} \ArcL{C}{D}{d} \\
\ArcL{A}{B}{a'} \ArcL{B}{D}{c'} \LArcR{C}{D}{b'} \EdgeL{B}{D}{c'} \LArcR{C}{D}{d'}
\%
\end{VCPicture}

The command \FinalState{\{(x,y)\}}{p} is equivalent to the sequence
\begin{VCPicture}{(0,-2)(6,2)}
\StateLineDouble \State{\{(x,y)\}}{p} \StateLineSimple \State{\{(x,y)\}}{p}
\end{VCPicture}
if the current implicit mode is \StateLineSimple, it is equivalent to \State{\{(x,y)\}}{p} in the other case.

\textbf{Hiding states}
\begin{itemize}
  \item \HideState \ShowState
\end{itemize}

Cancel the drawing of states. Active the drawing of states.

May be useful for drawing slides that will be overlayed.\footnote{We began to write the package when overhead projectors were still of common usage.}
\begin{VCPicture}{(0,-2)(6,2)}
% states
\HideState
\State[i]{(0,0)}{A} \State[t]{(6,0)}{C}
\ShowState
\State[q]{(3,0)}{B}
% initial-final
\Initial{A} \Final{C}
% transitions
\EdgeR{A}{B}{2b} \EdgeR{B}{C}{2b} \LArcL{A}{C}{b}
\LoopN{A}{a} \LoopS{A}{b} \LoopS[.5]{B}{2a+2b}
\LoopN[.75]{C}{4a} \LoopS[.75]{C}{4b}
% 
\end{VCPicture}

**Reverse transitions**

\ReverseArrow Exchange the origin and the end of a transition.

\StraightArrow Restore the normal style.

May be useful for modifying a part of a figure that had already been written.

\begin{VCPicture}{(0,-2)(6,2)}
% states
\State[A]{(0,0)}{A} \State[B]{(2,-2)}{B} \State[C]{(4,2)}{C} \State[D]{(6,0)}{D}
% transitions
\Final[s]{B} \ArcR{A}{B}{a} \ArcL{A}{C}{b} \ArcR{B}{D}{c} \ArcL{C}{D}{d}
\ReverseArrow \Initial{A} \Final[n]{C} \ArcL{A}{B}{a'} \EdgeR{A}{C}{b'} \EdgeL{B}{D}{c'} \EdgeL{C}{D}{d'}
% 
\end{VCPicture}

**Edge border**

\EdgeBorder Draw edges with a white border on each side.

\EdgeBorderOff Restore the normal style.

May be used to make more readable a picture in which many arrows cross each other. The *order* in which the edges are drawn becomes then meaningful.
Forth and back offset

Normally, an edge lays on the line that links the center of the start and end states. Which forbid to have two straight edges in opposite directions between states.

\ForthBackOffset
\RstEdgeOffset

Zigzag edges

\ZEdgeL\{Pos\}\{Start\}\{End\}\{Label\}
\ZEdgeR\{Pos\}\{Start\}\{End\}\{Label\}

Lining up

By default, labels of states are centered. When states are aligned, it may happen that this seems ungainly. The following command solves this problem.
\AlignedLabel Put labels of states on a virtual baseline.
\FloatingLabel Center the label of states vertically.

Default option is \FloatingLabel.

\begin{VCPicture}{(0,-4)(6,1)}
% states
\State[p]{(0,0)}{A} \State[b]{(3,0)}{B} \State[a]{(6,0)}{C}
\AlignedLabel
\State[p]{(0,-3)}{A1} \State[b]{(3,-3)}{B1}
\State[a]{(6,-3)}{C1}
%transitions
\EdgeL{A}{B}{a} \ArcL{B}{C}{b} \ArcL{C}{B}{c}
\EdgeL{A1}{B1}{a} \ArcL{B1}{C1}{b} \ArcL{C1}{B1}{c}
% \end{VCPicture}

The command \Point This commands is a direct translation of a command from PSTricks.

\Point{(x,y)}{Name} Sets a point with name Name at the coordinate (x, y).

This command does not draw any point. With PSTricks, and thus with V\textsc{auCan}\textsc{son-G}, any line or curve must be drawn between two predefined objects. That is the aim of \Point. For instance, initial and final arrows are drawn with this trick: any time a state is defined, eight points are silently and implicitly defined around it.

2.6 Writing composite labels

Some useful tools are proposed to write composite labels on arrows.

\IOL{Input}{Output} Produces the string Input|Output, useful when one draws transducers. This allows to modify the way it is done by redefining command \IOL.

\StackXLabels{Label1}{Label2} In these commands, X equals Two or Three. In the later case, commands have obviously three arguments.
\StackXLabelsP{Label1}{Label2} \StackXLabels gives a vertical arrangement of labels, whereas \StackXLabelsP inserts symbols + between them. \LineXLabelsP is the horizontal version of \StackXLabelsP; it can be redefined in order to change symbols, for instance.

The following automaton is the right sequential transducer that realizes the addition in base 2.
3 Composing a picture

The commands described above allow to define most of the figures representing automata in any textbook or technical paper. Before turning to more sophisticated commands that are used for less than 5 percent of the elements and that appear in less than 10 percent of the figures, let us turn to few commands that help in composing the figures.

3.1 Structuring the automaton: The \texttt{VCPut} command

In many automata, a same pattern in states, and also in edges, is reproduced several times in the figure. The idea is not to try to “program” these regular patterns but to allow the use of the “copy-and-paste” operations in the source file as much as possible, with minimum corrections afterwards. This will be achieved easily with the help of the command \texttt{VCPut}.

\begin{equation}
\texttt{VCPut}\{(x,y)\}{\{\text{instructions...}\}}
\end{equation}

Translates the result of the instructions by the vector \((x, y)\).

Note that this command is taken directly from \texttt{PSTricks} but uses \texttt{V\textsc{\textsc{a}}\textsc{\textsc{u}}\textsc{\textsc{c}}\textsc{\textsc{a}}\textsc{\textsc{n}}\textsc{\textsc{s}}\textsc{\textsc{on}}-G}'s syntax.

Of course the commands that are to be put as the second argument of \texttt{VCPut} are the defining states commands, with possibly other \texttt{VCPut} inside. The structuration of the figure \textit{via} the use of \texttt{VCPut} commands does not only save typing. It makes also easier the tuning of the figure, and the reusability of its parts.

In the example below, the two \texttt{VCPut} are used to place the two parts of the automaton.
The \texttt{\textbackslash VCPut} command can also be used to rotate a figure by using an optional parameter.

\texttt{\textbackslash VCPut}\{r\}(x,y)\{ instructions... \}

Rotate the result of the instructions with angle $r$ and center $(0,0)$ and then translate it by the vector $(x,y)$.

The default option is that labels of states (except those of \textit{variable states}) are put in a horizontal position (\texttt{\textbackslash RigidLabel}), but the command \texttt{\textbackslash SwivelLabel} makes them rotate with the figure.

One can notice that \texttt{\textbackslash RigidLabel} only acts on labels of states. To draw arcs with "rigid" labels, commands have to be written outside \texttt{\textbackslash VCPut}. We shall see later how to deal with the command \texttt{\textbackslash LoopL} in order to give any orientation to loops without using \texttt{\textbackslash VCPut}.
3.2 Using separate files

In order to use several times the same figure in the same or in different papers, or even, as we shall see in the next section, in two different contexts such as in a paper and in a slide for a talk, it is convenient to have the figure written into a file and imported in the main \TeX source file. This can be handled by the following macros.

\Call{FileName} \hspace{1cm} Calls the file FileName and apply the \Scale.
\SetVCDirectory{Directory} \hspace{1cm} Sets to Directory the directory where the file FileName will be taken by the macro \Call. Set to the current directory by default. Usual Unix syntax is used to name directories.
\ShowName \HideName \hspace{1cm} Sets or unsets a flag that controls the printing of FileName besides the figure drawn by \Call.

For the example below, the following lines have been saved in the file TwoStates.tex:

\begin{VCPicture}{(-1,-1)(4,1)}
\State[p]{(0,0)}{A} \State[q]{(3,0)}{B}\%
\ArcL(A){B}{b} \LArcL(B){A}{a} \LoopE{B}{a}
\end{VCPicture}

\Call{TwoStates} applies the same parameters to figures described in files as \Draw does.

Two macros that help placing the figure in the page, and the states in the figure.

\ShowFrame \HideFrame \hspace{1cm} Sets or unsets a flag that controls the drawing of the bounding box which is defined by the parameter of the \VCPicture environment.
\ShowGrid \HideGrid \hspace{1cm} Sets or unsets a flag that controls the drawing of a grid inside the bounding box which is defined by the parameter of the \VCPicture environment.

Default settings are \HideFrame and \HideGrid.

3.3 Frame and grid

Two macros that help placing the figure in the page, and the states in the figure.

\ShowFrame \HideFrame \hspace{1cm} Sets or unsets a flag that controls the drawing of the bounding box which is defined by the parameter of the \VCPicture environment.
\ShowGrid \HideGrid \hspace{1cm} Sets or unsets a flag that controls the drawing of a grid inside the bounding box which is defined by the parameter of the \VCPicture environment.

\ShowFrame \ShowGrid \Call{TwoStates}
3.4 The grid scale

This parameter controls the value of the PSTricks unit length. As values that define the size of states and the width of lines are given with fixed units (as cm or pt), they are not modified by this parameter. However, the scale of the grid on which they are put, \textit{i.e.} the distance between each other is modified by grid scale. This parameter is actually the first (optional) argument of \texttt{VCDraw} or \texttt{VCCall}.

The following example is the figure of Subsection 1.6 with another grid scale:

\begin{verbatim}
\VCDraw{.7}{}
\begin{VCPicture}{(0,-2)(6,2)}
% states
\State{i}{(0,0)}{A} \State{q}{(3,0)}{B} \State{t}{(6,0)}{C}
% initial-final
\Initial{A} \Final{C}
% transitions
\EdgeR{A}{B}{2b} \EdgeR{B}{C}{2b} \LArcL{A}{C}{b}
\LoopN{A}{a} \LoopS{A}{b} \LoopS{.5}{2a+2b}
\LoopN{C}{.75}{4a} \LoopS{.75}{4b}
\end{VCPicture}
\end{verbatim}

We can call \texttt{TwoStates} with another grid scale.

\begin{verbatim}
\ShowFrame \ShowGrid \VCCall{1.5}{TwoStates}
\end{verbatim}

4 Parametrized drawing commands

In the sequel, the letter \texttt{X} stands for either \texttt{L} or \texttt{R}.

4.1 Parametrized loops

One can draw loops around state with any direction.

\texttt{\LoopX[Pos]{Direction}{Name}{Label}} \quad \texttt{\CLoopX[Pos]{Direction}{Name}{Label}}

Draw a loop around the state \texttt{Name} in direction \texttt{Direction}, which is an angle with respect to east, with label \texttt{Label}.

As the label is placed outside the loop, \texttt{\LoopL} is reversed with respect to \texttt{\LoopR}. The opening of the loop is $60^\circ$ for the \texttt{\LoopX} commands, $44^\circ$ for the \texttt{\CLoopX} commands.
4.2 Parametrized arcs

\VArcX{arcangle=x,ncurv=v}(Start)(End)(Label) \quad \text{Draw an arc from Start to End with label Label, with a starting angle x and an eccentricity v.}

The starting angle is defined by the parameter arcangle=x and is measured, going to the left if X=L, to the right if X=R, from the line that goes from Start to End, called the base line. The parameter ncurv is the ‘eccentricity’ of the arc and some how measures the distance between the center of the arc and the base line as one can observe on the figure below.

The first (non optional) parameter of \VArcX contains PSTrick’s parameters, written with PSTrick’s syntax: they are separated by a comma and there must be no space inside the braces that contains them. All these parameters are optional, but this first parameter of \VArcX itself is not an optional argument, which means that this command must always have four arguments (and possibly an optional one):

\VArcL{arcangle=15}{A}{B}{x} \quad \text{are syntactically correct commands.}
\VArcL{arcangle=-120}{A}{B}{120}

Default value for arcangle is the same as for \LArC, whereas the default ncurv value is 1.
4.3 General curve

\texttt{\textbackslash CurveX[Pos]{angleA=x,angleB=y,ncurv=v}{Start}{End}{Label}}

Draw a curve from \textit{Start} to \textit{End} with label \textit{Label}, starting with an angle \textit{x}, arriving with an angle \textit{y} and with an eccentricity \textit{v}.

A (Bézier) curve, computed by \texttt{PSTricks} with the three parameters: the \textit{starting angle} fixed by \texttt{angleA=x}, the ending angle fixed by \texttt{angleB=y}, and the eccentricity \texttt{ncurv=v}.

The two angles are measured, as trigonometric angles are, from the East direction and counterclockwise. The syntax of these parameters is \texttt{PSTricks}'s and they are optional, as in \texttt{\textbackslash ArcX}. Default values for \texttt{angleA} and \texttt{angleB} are respectively 0 and 180; \texttt{ncurv} is fixed by default to 1.

\[
\begin{VCPicture}{(-4,-4)(4,4)}
%states
\State[A]{(-3,0)}{A}
\State[B]{(0,0)}{B}
\State[C]{(3,0)}{C}
%transitions
\VCurveL{angleA=90,angleB=135}{A}{B}{1}
\VCurveR{.5}{angleA=90,angleB=135,ncurv=.5}{A}{B}{0.5}
\VCurveL{angleA=90,angleB=135,ncurv=2}{A}{B}{2}
\VCurveR{.6}{angleA=-80,angleB=175}{A}{B}{b}
\VCurveR{.6}{angleA=-80,angleB=175}{A}{C}{c}
\VCurveL{angleB=20,angleB=160}{B}{C}{d}
\end{VCPicture}

4.4 Transitions labelling

\textbf{Initial and final arrows with a label} When one deals with tranducers, or automata with multiplicity, one may need to put some labels on initial or final arrows.

\texttt{\InitialL[Pos]{Dir}{Name}{Label}} \quad \texttt{\InitialR[Pos]{Dir}{Name}{Label}}

Makes the state \textit{Name} initial and writes \textit{Label} on the left or the right size of the incoming arrow.

\texttt{\FinalL[Pos]{Dir}{Name}{Label}} \quad \texttt{\FinalR[Pos]{Dir}{Name}{Label}}

Makes the state \textit{Name} final and writes \textit{Label} on the left or the right size of the outgoing arrow.

Here, the parameter \textit{Dir} is not optional. Whereas, there is an optional parameter \textit{Pos} that defines the position of the label on the arrow. Its default values is fixed to 0.1 (resp. 0.9) for initial (resp. final) arrows. The label is thus written far enough of the state.

\textbf{Independant labelling of transitions} There exists commands to label transitions that have not been labeled yet or to put a second label on them. These commands must directly follow the definition of the transition that will be labeled.
\texttt{\textbackslash LabelL[Pos]\{Label\}} \texttt{\textbackslash LabelR[Pos]\{Label\}} \texttt{Puts a label on the previous transition. The default position of the label is the same as for edges.}

\begin{VCPicture}{(-3,-2)(3,2)}
\State[A]{(-2,1)}{A} \State[B]{(2,1)}{B} \\
\State[C]{(-2,-1)}{C} \State[D]{(2,-1)}{D} \\
% \\
\EdgeL{A}{B}{\varphi(u)=3} \LabelR{\varphi(u)=3} \\
\LArcR{D}{C}{} \LabelL{v} \LabelL{v} \\
\VCurveL[.3]{angleA=45,angleB=-45}{B}{D}{a} \\
\LabelL[.5]{b} \LabelL[.7]{c} \\
\end{VCPicture}

4.5 Miscellaneous

Loops on variable states Instead of using \texttt{\textbackslash VarLoopN} or \texttt{\textbackslash VarLoopS}, one can modify parameters of loops to draw, for instance, many loops on variable states. As usual, these loops may be drawn on the south or north side of the states only.

\texttt{\textbackslash VarLoopOn} \texttt{\textbackslash VarLoopOff} \texttt{Modifies parameters in order to draw loops around variable states.} \texttt{\textbackslash LoopVarN} \texttt{\textbackslash LoopVarOff} \texttt{Restore the normal style of loops.}

Command \texttt{\textbackslash LoopVarN} is equivalent to the sequence \texttt{\textbackslash LoopVarOn \textbackslash LoopN \textbackslash LoopVarOff}.

Saving memory In default mode, the call to a \texttt{\textbackslash State} command defines eight additional points, that allow then to draw initial and final arrows in the prescribed compass direction. In case of large figures, with many states, this ought to lead to an overflow of \TeX memory. This is the reason why one can define states without any additional point.

\texttt{\textbackslash PlainState} \texttt{Draw states with no additional points.} \texttt{\textbackslash FullState} \texttt{Draw states with eight points to draw initial or final arrows.}

Of course, when in \texttt{\textbackslash PlainState} mode, one can define states with additional points.

\texttt{\textbackslash IFState\{Label\}(x,y)\{Name\}} \texttt{Draws a state with four points \texttt{(n, s, e, w)}.} \texttt{\textbackslash IFXState\{Label\}(x,y)\{Name\}} \texttt{Draws a state with eight points \texttt{(n, s, e, w, ne, se, nw, sw)}.}

In \texttt{\textbackslash PlainState} mode, one should not apply an \texttt{\textbackslash Initial} or a \texttt{\textbackslash Final} command to a state defined by \texttt{\textbackslash State}. Likewise, these commands can be only used for \texttt{\textbackslash IFState} with directions \texttt{n, s, e} and \texttt{w}.

Note that even when in \texttt{\textbackslash FullState} mode, variable states defined by \texttt{\textbackslash StateVar} have the same four additional points as \texttt{\textbackslash IFState}. 
5 Modifying and setting the parameters

The reason why the commands in VĂUCĂNSON-G are simple and concise is that all the parameters that control the geometry (size, angles, ...) and the drawing (line width, line style, line color, ...) of the states and transitions in an automaton are preset by VĂUCĂNSON-G and can be kept implicit. But there are two instances where a user may want to modify these preset values.

The first case is when one needs to tune a parameter in order to make the figure more readable, or nicer. Such a case has already been considered in sections 1.3, 1.4 and 1.5, where the parameter Pos that controls the position of the label of a transition along the edge has been modified with an optional argument. The parameters that will be considered in this section are not optional parameters of already defined commands but parameters that are set with their own commands.

The other case where a user will change the implicit parameters of a figure or, more likely, of a set of figures, is when he (or she) will be in the position to change completely the “style” of the drawing. For instance, a same set of figures may be used in a paper and in a series of slides for the presentation of the paper. Of course, the figures have to be set at a larger scale, probably in colour, and possibly with different relative sizes of the elements.

We begin with the description of the options for the latter case, which is by far simpler. We then explain the syntax convention taken to coin the names of the many commands that control the many parameters. Then comes the list of these parameters, for scaling first, then for states, for transitions, and finally for the fine tuning of some details of the drawing.

5.1 Style files

The package VĂUCĂNSON-G is loaded with default values that are all set up in the file VCPref-default.tex. The package comes with two other styles, in particular one for drawing slides. The normal way of using the different style files is to call the wrapper vaucanson-g with an option:

\usepackage[option]{vaucanson-g}

The default option is default; the other possible options are slides, beamer and mystyle.

Although it is not the normal usage, it is also possible to change style in the course of the text by loading a style file with an \input command. In this case, it is recommended to reset all parameters by calling VCPref-default.tex.
By editing the files \texttt{VCPref-mystyle}, \texttt{VCPref-beamer} or \texttt{VCPref-slides}, and giving the parameters the values he (or she) thinks more appropriate, a user may easily build his (or her) own style for drawing automata. Modifying directly \texttt{VCPref-default}, although possible, is not necessarily a good idea. By cut, editing and paste, it is not difficult create many style files, and even, with an easy modification of the wrapper itself to make them possible options. The content of these style files are described in the following subsections. \texttt{VCPref-slides}, \texttt{VCPref-beamer} and \texttt{VCPref-mystyle} are given at Appendix B.

### 5.2 Syntax of parameter settings

The set of values of all parameters defines a style of figures. Most of parameters, \textit{e.g.} \texttt{\StateLineColor} which defines the color of the line that delimits a state, may be given a temporary value, within a figure, without changing the style. These parameters are dealt with by means of 3 different commands:

\begin{itemize}
  \item \texttt{\ChgParam{ Val }} \quad \text{Gives the new value \texttt{Val} to \texttt{Param}.}
  \item \texttt{\RstParam} \quad \text{Restores the style value in \texttt{Param}.}
  \item \texttt{\SetParam{ Val }} \quad \text{Sets the style value for \texttt{Param} to \texttt{Val}.}
\end{itemize}

The command \texttt{\SetParam} can be called outside a \texttt{VCPicture} environment, in which case it will be effective in all subsequent figures, until a new setting of the same parameter. Its natural place, if it is considered as a modification of the style, is in the preamble, after the call to the package \texttt{VAUCANSON-G}. If the command \texttt{\SetParam} is called inside a \texttt{VCPicture} environment, its effect will be restricted to that figure.

Such parameters will be called \textit{adjustable parameters} in the sequel.
Note that in the case where \texttt{\textbackslash Param} is a value (a \textit{length} or a \textit{scale}), the value of \texttt{\textbackslash Val} in \texttt{\textbackslash ChgParam\{\textbackslash Val\}} is a \textit{coefficient} which is applied to the style value of \texttt{\textbackslash Param} defined with \texttt{\textbackslash SetParam\{\textbackslash Val\}}. Therefore, in this case, \texttt{\textbackslash RstParam} is equivalent to \texttt{\textbackslash ChgParam\{1\}}.

The other parameters, as those that define the “dimmed” style, are called \textit{preset parameters}.

There is finally a class of parameters that fall in between the adjustable and the preset ones, those for which several values are “preset” but that can be given other values. These parameters will be called \textit{multivalued preset parameters}.

5.3 Scale parameter (Multivalued preset parameter)

\texttt{\textbackslash FixVCScale\{\textbackslash Scale\}}

\texttt{\FixVCScale\{\textbackslash Scale\}} \texttt{Sets} \texttt{\textbackslash VCScale} to \texttt{\textbackslash Scale}.

The commands \texttt{\textbackslash MediumPicture}, \texttt{\textbackslash SmallPicture}, \texttt{\textbackslash TinyPicture} and \texttt{\LargePicture} set \texttt{\textbackslash VCScale} to the values \texttt{\textbackslash MediumScale}, \texttt{\textbackslash SmallScale}, \texttt{\textbackslash TinyScale} and \texttt{\LargeScale} respectively. Their values are

<table>
<thead>
<tr>
<th>\texttt{\TinyScale}</th>
<th>\texttt{\SmallScale}</th>
<th>\texttt{\MediumScale}</th>
<th>\texttt{\LargeScale}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.42</td>
<td>0.5</td>
<td>0.6</td>
<td>0.85</td>
</tr>
</tbody>
</table>

These preset values can be modified by a \texttt{\textbackslash renewcommand}.

5.4 State parameters

State diameter (Multivalued preset parameter) This parameter controls, \textit{via} the commands \texttt{\textbackslash MediumState}, \texttt{\textbackslash SmallState} or \texttt{\LargeState} the value of a number of other preset parameters.

\texttt{\textbackslash FixStateDiameter\{\textbackslash Diam\}}

\texttt{\FixStateDiameter\{\textbackslash Diam\}} \texttt{Sets} the state diameter to \texttt{\textbackslash Diam} (\texttt{\textbackslash Diam} is a \textit{length}) until the next call to a \texttt{\ZState} command, or to the next figure if called inside a \texttt{\VCPicture}.

Preset diameter sizes are:

<table>
<thead>
<tr>
<th>\texttt{\VerySmallState}</th>
<th>\texttt{\SmallState}</th>
<th>\texttt{\MediumState}</th>
<th>\texttt{\LargeState}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3cm</td>
<td>0.6cm</td>
<td>0.9cm</td>
<td>1.2cm</td>
</tr>
</tbody>
</table>

These values can be modified by a \texttt{\textbackslash renewcommand}. However, as commands that call these parameters deal also with other parameters as the eccentricity of loops or the size of initial and final arrows, changing these values without modifying the other ones can lead to an ungainly result.

The shape of a normalized automaton can be drawn in the following way:
\begin{VCPicture}{(-5,-3)(5,3)}
\FixStateDiameter{5cm} \ChgStateLabelScale{3}
\State[\{cal A\}]{(0,0)}{A}
\MediumState \RstStateLabelScale
\State[i]{(-4,0)}{I} \State[t]{(4,0)}{T}
\VSState{(-1,-1)}{P1} \VSState{(1,-1)}{Q1}
\VSState{(-1,1)}{P2} \VSState{(1,1)}{Q2}
\VSState{(-1.4,0)}{P3} \VSState{(1.4,0)}{Q3}
\Initial{I} \Final{T}
\EdgeL{I}{P1}{}
\EdgeL{I}{P2}{}
\EdgeL{I}{P3}{}
\EdgeL{Q1}{T}{}
\EdgeL{Q2}{T}{}
\EdgeL{Q3}{T}{}
\end{VCPicture}

State adjustable parameters Each following parameter can be modified by the command \ChgParameter. The style value can be restored by \RstParameter and one can set the style value with \SetParameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>StateLineStyle</td>
<td>solid</td>
<td>dashed, dotted, none</td>
</tr>
<tr>
<td>StateLineWidth</td>
<td>1.8pt</td>
<td></td>
</tr>
<tr>
<td>StateLineColor</td>
<td>black</td>
<td>red, lightgray, ...</td>
</tr>
<tr>
<td>StateLabelColor</td>
<td>black</td>
<td>blue, Salmon, ...</td>
</tr>
<tr>
<td>StateLabelScale</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>StateFillStatus</td>
<td>solid</td>
<td>vlines, hlines, crosshatch, none</td>
</tr>
<tr>
<td>StateFillColor</td>
<td>white</td>
<td>green, Purple, ...</td>
</tr>
</tbody>
</table>

As it has been previously noticed, the argument of \ChgStateLineWidth is not a length but a coefficient applied to the style width. For instance, with the default value of StateLineWidth, after \ChgStateLineWidth{2} edges will be drawn with a 3.6pt width.

When pictures are drawn to scale \VCScale that is set to 0.6, printing state labels with scale 1.7 makes them to be as large as characters of the text. As for line width, \ChgStateLabelScale{.5} sets the label scale to 0.85 that make then twice smaller than characters of the text (because of \VCScale).
**State preset parameters**  They deal with the definition of “dimmed” states.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DimStateLineStyle</td>
<td>solid</td>
</tr>
<tr>
<td>DimStateLineColor</td>
<td>gray</td>
</tr>
<tr>
<td>DimStateLineCoef</td>
<td>1</td>
</tr>
<tr>
<td>DimStateLabelColor</td>
<td>gray</td>
</tr>
<tr>
<td>DimStateFillColor</td>
<td>white</td>
</tr>
</tbody>
</table>

*DimStateLineCoef* is a coefficient that is applied to *StateLineWidth* in drawing dimmed states. It might be larger than 1 to compensate optical effects when *DimStateLineStyle* is dashed or dotted.

The style of dimmed states can be fixed by the command \FixDimState{LineStyle}{LineColor}{LineCoef}{LabelColor}{FillColor}.

In mode *StateLineDouble* or when one uses the command \FinalState, two parameters give the width of each of both lines and the distance between them. These values are coefficients that are applied to *StateLineWidth*.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>StateLineDblCoef</td>
<td>0.6</td>
</tr>
<tr>
<td>StateLineDblSep</td>
<td>0.4</td>
</tr>
</tbody>
</table>

These values can be fixed by the command \FixStateLineDouble{Coef}{Sep}.

### 5.5 Transition parameters

**Edge adjustable parameters**  Each following parameter can be modified by the command \ChgParameter. The style value can be restored by \RstParameter and one can set the style value with \SetParameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EdgeLineStyle</td>
<td>solid</td>
<td>dashed, dotted, none</td>
</tr>
<tr>
<td>EdgeLineWidth</td>
<td>1pt</td>
<td></td>
</tr>
<tr>
<td>EdgeLineColor</td>
<td>black</td>
<td>many colours</td>
</tr>
<tr>
<td>EdgeLineDblStatus</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>EdgeNodeSep</td>
<td>0pt</td>
<td></td>
</tr>
<tr>
<td>EdgeLabelColor</td>
<td>black</td>
<td>many colours</td>
</tr>
<tr>
<td>EdgeLabelScale</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>EdgeLabelSep</td>
<td>3.5pt</td>
<td></td>
</tr>
</tbody>
</table>
As for states, the argument of \texttt{ChgEdgeLineWidth} is a coefficient applied to the style line width.

The parameter \texttt{EdgeNodeSep} defines a distance between the beginning, and the end, of a transition and the state it starts from and arrives to. The default value is 0pt, that is, no distance at all; but it may be the case, especially when there is no line around the state that one may want to increase the distance between the transition and the label of the state.

The parameter \texttt{EdgeLabelSep} controls the distance between the box that contains the label of the transition (built by \LaTeX) and the transition.

\textbf{Edge preset parameters} A first class of preset parameters deals with the definition of “dimmed” edges.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DimEdgeLineStyle</td>
<td>solid</td>
</tr>
<tr>
<td>DimEdgeLineColor</td>
<td>gray</td>
</tr>
<tr>
<td>DimEdgeLineCoef</td>
<td>1.2</td>
</tr>
<tr>
<td>DimEdgeLabelColor</td>
<td>gray</td>
</tr>
</tbody>
</table>

The style of dimmed edges can be fixed by the command \texttt{\texttt{FixDimEdge}{LineStyle}{LineColor}{LineCoef}{LabelColor}}.

The style of the border of edges (in mode \texttt{EdgeBorder}), is defined by two parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EdgeLineBorderCoef</td>
<td>2</td>
</tr>
<tr>
<td>EdgeLineBorderColor</td>
<td>white</td>
</tr>
</tbody>
</table>

\texttt{EdgeLineBorderCoef} is a coefficient applied to \texttt{EdgeLineWidth}, that gives the width of the border.

The style of borders of edges can be fixed by the command \texttt{\texttt{FixEdgeBorder}{Coef}{Color}}.

The width of the double line of (in mode \texttt{EdgeLineDouble}), is defined by two parameters, that give the width of each of both lines and the distance between them. These values are coefficients that are applied to \texttt{EdgeLineWidth}.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EdgeLineDblCoef</td>
<td>0.5</td>
</tr>
<tr>
<td>EdgeLineDblSep</td>
<td>0.6</td>
</tr>
</tbody>
</table>

These values can be fixed by the command \texttt{\texttt{FixEdgeLineDouble}{Coef}{Sep}}.
5.6 Advanced parameters

These parameters can be changed by using the usual command \texttt{\renewcommand} (or \texttt{\setlength}, when the parameter is a length).

As it has already be mentionned, the preset commands that set the diameter of states control parameters for drawing edges (the length of initial and final arrows or the curvature of loops). Their values are:

<table>
<thead>
<tr>
<th>X</th>
<th>ArrowOnX</th>
<th>LoopOnX</th>
<th>CLoopOnX</th>
</tr>
</thead>
<tbody>
<tr>
<td>LargeState</td>
<td>1.3</td>
<td>5.8</td>
<td>6</td>
</tr>
<tr>
<td>MediumState</td>
<td>1.5</td>
<td>5.8</td>
<td>8</td>
</tr>
<tr>
<td>SmallState</td>
<td>1.7</td>
<td>5.8</td>
<td>12</td>
</tr>
<tr>
<td>VariableState</td>
<td>5.1</td>
<td>5.2</td>
<td></td>
</tr>
<tr>
<td>VerySmallState</td>
<td>5</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

\textit{ArrowOnX} is the coefficient that is applied to the radius of the state to obtain the length of initial or final arrows. \textit{LoopOnX} and \textit{CLoopOnX} is the eccentricity of loops and closed loops respectively.

Another parameter determines whether the dimension of states refer to the middle, the inner or the outer side of the boundary. There exist one parameter for classical states and one for states with double line:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>\StateDimen</td>
<td>outer</td>
</tr>
<tr>
<td>\StateDblDimen</td>
<td>middle</td>
</tr>
</tbody>
</table>

The possible values are \texttt{inner}, \texttt{outer} or \texttt{middle}.

\begin{VCPicture}{(-1,-1)(6,2)}
\ChgStateLineWidth{5}
\FixStateDiameter{1cm}
\renewcommand{\StateDimen}{inner}
\State{(0.5,0.5)}{I}
\renewcommand{\StateDimen}{middle}
\State{(2.5,0.5)}{M}
\renewcommand{\StateDimen}{outer}
\State{(4.5,0.5)}{O}
\end{VCPicture}
\HideGrid

\textbf{Transition adjustable geometric parameters} All these parameters are adjustable. That means there exist commands as \texttt{\SetParameter}, \texttt{\ChgParameter} and \texttt{\RstParameter}.

Arrows between states are characterized by some adjustable parameters:
Except the curvature that depends on the size of states, parameters that define loop, the degree of opening and the offset, are adjustable parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EdgeArrowStyle</td>
<td>-&gt;</td>
</tr>
<tr>
<td>EdgeArrowWidth</td>
<td>5pt</td>
</tr>
<tr>
<td>EdgeArrowLengthCoeff</td>
<td>1.4</td>
</tr>
<tr>
<td>EdgeArrowInsetCoeff</td>
<td>0.1</td>
</tr>
</tbody>
</table>

The length and the inset of arrows are given by coefficients that are applied to the width. There exists some other (non adjustable) parameters to define arrows in special cases: \( \text{\textbackslash{}EdgeArrowRevStyle} \) is equal to \(<->\), to draw transition in reverse mode; for edges with double lines, the size of arrows is a little bit more larger. It is defined by \( \text{\textbackslash{}EdgeDblArrowWidth} \) and \( \text{\textbackslash{}EdgeDblArrowLengthCoeff} \) that are respectively equal to 5.5pt and 1.7.

Zigzag edges are characterized by some adjustable parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZZSize</td>
<td>0.9cm</td>
</tr>
<tr>
<td>ZZLineWidth</td>
<td>1.7</td>
</tr>
</tbody>
</table>

\( \text{ZZSize} \) is the apparent diameter of the zigzag and \( \text{ZZLineWidth} \) is a coefficient that is applied to \( \text{EdgeLineWidth} \) and gives the width of the line.
**Transition preset parameters**

The default position of labels on transitions depends on the type of transition. For instance, on edges, this position is given by the variable \\
\texttt{\textbackslash EdgeLabelPosit} that is equal to 0.45. The first column of the following array gives these values for every transition.

<table>
<thead>
<tr>
<th>X</th>
<th>\texttt{\textbackslash LabelPosit}</th>
<th>\texttt{\textbackslash LabelRevPosit}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>.45</td>
<td>.55</td>
</tr>
<tr>
<td>Arc</td>
<td>.4</td>
<td>.6</td>
</tr>
<tr>
<td>LArc</td>
<td>.4</td>
<td>.6</td>
</tr>
<tr>
<td>Loop</td>
<td>.25</td>
<td>.75</td>
</tr>
<tr>
<td>CLoop</td>
<td>.25</td>
<td>.75</td>
</tr>
<tr>
<td>Init</td>
<td>.1</td>
<td>.9</td>
</tr>
<tr>
<td>Final</td>
<td>.9</td>
<td>.1</td>
</tr>
</tbody>
</table>

When the reverse mode is on, the position of label is given by variables that correspond to the second column, as \texttt{\textbackslash EdgeLabelRevPosit} for instance. For the same type of transition, the sum of both variables that defined position for straight and reverse arrows should be equal to 1 (if one wants to keep the same appearance when using the ReverseArrow command).

These parameters can be modified with a \texttt{\textbackslash renewcommand}.

---

**6 Using “plain” PSTricks commands**

\texttt{\textbackslash VauCanSonG} is a set of macros based on \texttt{PSTricks}. It is therefore fully compatible with \texttt{PSTricks} instructions. Actually, the \texttt{VCPicture} environment is a specialization of the \texttt{pspicture} environment; more, every state in \texttt{\textbackslash VauCanSonG} is a \texttt{PSTricks} node and every transition (loop, edge, arc) is a node connection.

The following example shows how \texttt{\textbackslash VauCanSonG} commands can be mixed with plain \texttt{PSTricks} commands. For further explanation on \texttt{PSTricks}, we refer to [1, 3, 4, 6].

```latex
\begin{VCPicture}{(-7.5,-1)(3,2)}
\rput(-5,0){\rnode{X}{\psframebox{$\bigoplus$}}} \rput(.5,0){\rnode{Y}{\psframebox{$\bigoplus$}}} \State[1]{(-3.5,0)}{A} \State[0]{(-1,0)}{B} \State[0]{(2,0)}{C} \Point{(-7,0)}{S} \EdgeL{S}{X}{0011} \ncangles[angleB=90, armB=1cm]{C}{X} \ncangles[angleB=90, armB=1cm]{C}{Y} \end{VCPicture}
```

\texttt{\begin{VCPicture}{(-7.5,-1)(3,2)}
\rput(-5,0){\rnode{X}{\psframebox{$\bigoplus$}}} \rput(.5,0){\rnode{Y}{\psframebox{$\bigoplus$}}} \State[1]{(-3.5,0)}{A} \State[0]{(-1,0)}{B} \State[0]{(2,0)}{C} \Point{(-7,0)}{S} \EdgeL{S}{X}{0011} \EdgeL{X}{A}{\} \EdgeL{A}{B}{\} \EdgeL{B}{Y}{\} \EdgeL{Y}{C}{\} \ncangles[angleB=90, armB=1cm]{C}{X}{\} \ncangles[angleB=90, armB=1cm]{C}{Y}{\} \end{VCPicture}
}

The shif register for $1 + x^2 + x^3$
A Version history

A first version of the package, which would be numbered 0.1, was called Vaucanson and used by the authors in their papers between 2000 and 2003. But, at the same time, the authors were involved, in cooperation with the LRDE at EPITA, in the design of a C++ platform for programming automata, for which it was decided that there would be no better name than Vaucanson as well. We thus turned the name of the package for drawing automata to VAUCANSON-G.

Version 0.2 was issued in May 2003 and made available on the web together with this user’s manual first version. A series of bug corrections and slight improvements made us switch at some point to version 0.3.

Just before sending the package to CTAN, we undertook a full review of the code and updated the manual accordingly. We call it version 0.4. It is the last release before a completely new version of the package, with many more options, which is currently under work.
B Style files

B.1 slides style

\renewcommand{\LargeScale}{1.5}
\renewcommand{\MediumScale}{1.16}
\renewcommand{\SmallScale}{0.92}
\renewcommand{\TinyScale}{0.75}

% State aspect
\SetStateLineColor{blue}
\SetStateLineWidth{2pt}
\SetStateFillColor{Cyan}
\SetStateLabelColor{Sepia}
\FixDimState{solid}{YellowOrange}{1}{YellowOrange}{white}

% Edge aspect
\SetEdgeLineColor{Mahogany}
\SetEdgeLineWidth{1.2pt}
\SetEdgeLabelColor{OliveGreen}
\FixDimEdge{dashed}{1.2}{Apricot}{Apricot}
B.2 beamer style

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Package ‘Vaucanson-G’ version 0.4
%% This is file ‘VCPref-slides’.
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Scales --- slides settings
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\renewcommand{\LargeScale}{0.85}
\renewcommand{\MediumScale}{0.6}
\renewcommand{\SmallScale}{0.4}
\renewcommand{\TinyScale}{0.30}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% State aspect
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\SetStateLineColor{blue}
\SetStateLineWidth{2pt}
\SetStateFillColor{Cyan}
\SetStateLabelColor{Sepia}
\FixDimState{solid}{YellowOrange}{1}{YellowOrange}{white}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Edge aspect
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\SetEdgeLineColor{Mahogany}
\SetEdgeLineWidth{1.2pt}
\SetEdgeLabelColor{OliveGreen}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Dimmed edges
\FixDimEdge{dashed}{1.2}{Apricot}{Apricot}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B.3 mystyle style

This file is given only on the purpose that it can be edited by the user and called by the option mystyle of the wrapper (and this is the reason why the name VCPref-mystyle.tex of this file should not be changed).

The actual values given in this file as an example provide a style which mimics the way automata are drawn with GasTeX [2] as can be witnessed below.

\begin{VCPicture}{(0,-2)(6,2)}
% states
\State[A]{(0,0)} \State[B]{(2,-2)} \FinalState[C]{(4,2)} \State[D]{(6,0)}
% initial-final
\Initial{A} \Final{s}{B}
% transitions
\ArcR{A}{B}{a} \ArcL{A}{C}{b} \LArcR{B}{D}{c} \LArcL{C}{D}{d} \ArcL{A}{B}{a'} \EdgeLineDouble \EdgeR{A}{C}{b'}
\EdgeLineSimple \EdgeL{B}{D}{c'} \LArcR{C}{D}{d'}
% \end{VCPicture}
References


http://www.lsv.ens-cachan.fr/~gastin/gastex/gastex.html


2007.

Some examples of articles or slides that contain automata drawn with \texttt{\LaTeX\textsc{G}} can be found at the following URLs:

- http://www.enst.fr/~jsaka/
- http://igm.univ-mlv.fr/~lombardy/