Automatica

Lars Nagel

Technische Universität München
lars.nagel@stud.tu-muenchen.de

Zusammenfassung Diese Bibliothek dient als Hilfsmittel beim Zeichnen endlicher Automaten. Die in Automatica zusammengefassten Makros sind mit METAPOST programmiert worden und können über den `input`-Befehl zu jeder METAPOST-Datei dazugebunden werden (`input automat.mp`).

Inhaltsverzeichnis

1 Einleitung

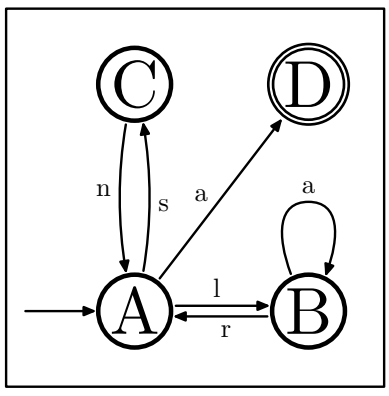
Automatica ist eine Sammlung von METAPOST-Makros, die beim Zeichnen endlicher Automaten helfen soll. Es lassen sich aber auch andere Graphen, z. B. Petri-Netze, damit zeichnen. Die in dieser Dokumentation aufgeführten METAPOST-Variablentypen (`numeric`, `pair`, `string`, `color`, `boolean`, `picture` und `pen`) sind in „A User's Manual for MetaPost“ [Hob92] erklärt.

1.1 Einordnung in die LaTeX-Welt

Anders als bei ähnlichen Makro-Paketen wie „GasTeX“ [Gas03] oder „Vaucanson-G“ [LS03] werden die Automatica-Makros in METAPOST und nicht direkt in LaTeX verwendet. Das hat den Vorteil, dass der Benutzer beim Zeichnen auch auf die METAPOST-Befehle zurückgreifen kann. Automatica ist wie „Vaucanson-G“ auf endliche Automaten spezialisiert, wobei der Befehlssatz verglichen mit „Vaucanson-G“ nur unwesentlich kleiner ist. Ein Vorteil gegenüber „Vaucanson-G“ ist, dass rechteckige Zustände (bzw. Transitionen) gezeichnet werden können, was auch Petri-Netze ermöglicht. Allerdings reicht es in dieser Hinsicht nicht an „GasTeX“ heran, mit dem sowohl Petri-Netze als auch endliche Automaten gleichermaßen gut gezeichnet werden können. Nachteilig an „GasTeX“ ist jedoch, dass die Befehle relativ lang und kompliziert sind und dass es keine Dokumentation zum Paket gibt. Die wesentliche Idee bei Automatica ist, dass es für einfache Dinge einfache Befehle bereit hält, dass aber auch Kompliziertes möglich ist und im Ausnahmefall noch METAPOST zur Verfügung steht.

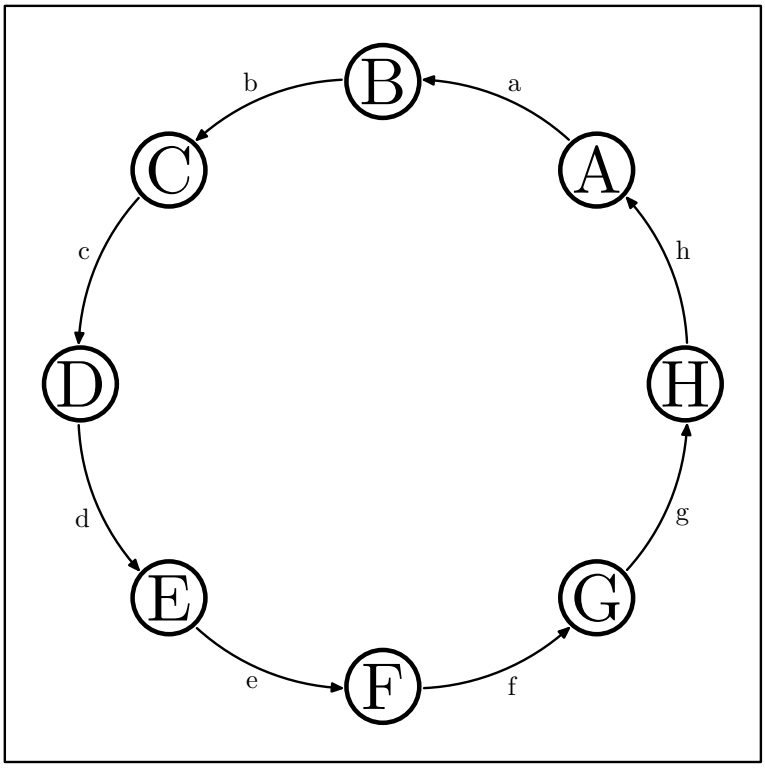
1.2 Einführende Beispiele

Vor der eigentlichen Anleitung sollen drei einfache Beispiele demonstrieren, wie Automatica verwendet wird. Zuerst ein einfacher endlicher Automat mit einem Startzustand „A“ und einem Endzustand „D“. (In den folgenden Beispielen werden `beginfig(...);` und `endfig;` weggelassen.)



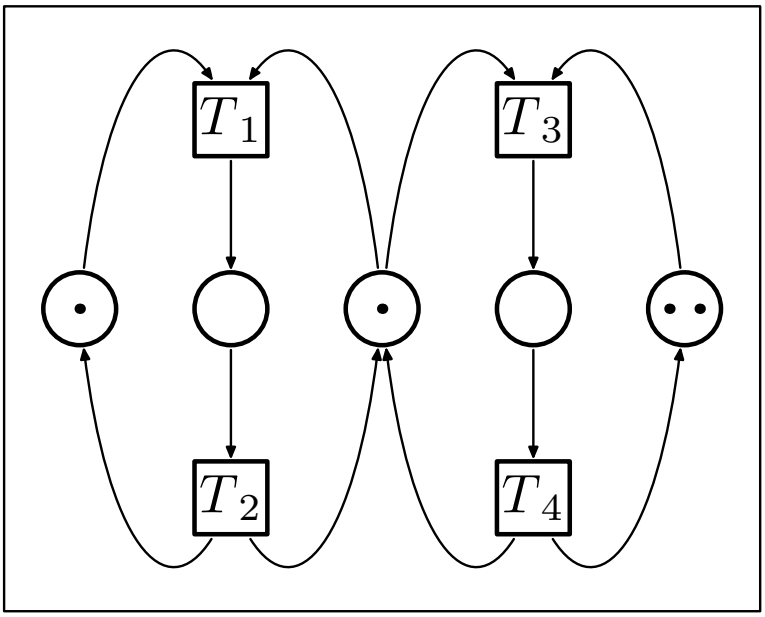

```
beginfig(23);
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm)); ShowFrame;
%%% Zustaende
State.A((1.7cm,1cm),"A"); Initial.A("w");
State.B((4cm, 1cm), "B");
State.C((1.7cm, 4cm), "C");
FinalState.D((4cm, 4cm), "D");
%%% Transitionen
ForthBackOffset; EdgeR(A,B,"l",0.45);
EdgeR(B,A,"r",0.45); RstEdgeOffset;
ArcR(A,C,"s",0.45); ArcR(C,A,"n",0.45);
EdgeL(A,D,"a",0.45); LoopN(B,"a",0.5);
EndAutomata;
endfig;
```

Im zweiten Beispiel werden METAPOST-Befehle (für die Schleife, für Sinus und Kosinus) verwendet, um die Koordinaten der Zustände zu berechnen, die in gleichen Abständen auf einem Kreis angeordnet werden.




```
BeginAutomata;
SetFrame((-5cm, -5cm), (5cm, 5cm)); ShowFrame;
pair q[];
for i=1 upto 8:
  q[i]:=(cosd(i*45)*4cm,sind(i*45)*4cm); endfor
%%% Zustaende
State.A(q[1], "A"); State.B(q[2], "B");
State.C(q[3], "C"); State.D(q[4], "D");
State.E(q[5], "E"); State.F(q[6], "F");
State.G(q[7], "G"); State.H(q[8], "H");
%%% Transitionen
LArcR(A, B, "a", 0.5); LArcR(B, C, "b", 0.5);
LArcR(C, D, "c", 0.5); LArcR(D, E, "d", 0.5);
LArcR(E, F, "e", 0.5); LArcR(F, G, "f", 0.5);
LArcR(G, H, "g", 0.5); LArcR(H, A, "h", 0.5);
EndAutomata;
```

Das dritte Beispiel zeigt ein Petri-Netz mit Stellen und Transitionen. Die Marken werden mit dem `draw`-Befehl von METAPOST in die Stellen gezeichnet.




```
BeginAutomata;
SetFrame((0, 0), (10cm, 8cm)); ShowFrame;
%%% Zustaende und "Transitionen"
State.A((1cm, 4cm), ""); State.B((3cm, 4cm), "");
State.C((5cm, 4cm), ""); State.D((7cm, 4cm), "");
State.E((9cm, 4cm), "");
pickup pencircle scaled 4;
draw((1cm, 4cm)); draw((5cm, 4cm));
draw((8.8cm, 4cm)); draw((9.2cm, 4cm));
pickup pencircle scaled 1;
QState.F((3cm, 6.5cm), btex $T{\sb 1}$ etex);
QState.G((3cm, 1.5cm), btex $T{\sb 2}$ etex);
QState.H((7cm, 6.5cm), btex $T{\sb 3}$ etex);
QState.I((7cm, 1.5cm), btex $T{\sb 4}$ etex);
%%% Pfeile
EdgeR(F, B, "", 0.5); EdgeR(B, G, "", 0.5);
EdgeR(H, D, "", 0.5); EdgeR(D, I, "", 0.5);
VCurveL(A,F, 85,110,1.5,"",0.5); VCurveL(C,F, 95, 70,1.5,"",0.5);
VCurveL(G,A,250,275,1.5,"",0.5); VCurveL(G,C,290,265,1.5,"",0.5);
VCurveL(C,H, 85,110,1.5,"",0.5); VCurveL(E,H, 95, 70,1.5,"",0.5);
VCurveL(I,C,250,275,1.5,"",0.5); VCurveL(I,E,290,265,1.5,"",0.5);
EndAutomata;
```

2 Zeichenumgebung

2.1 BeginAutomata und EndAutomata

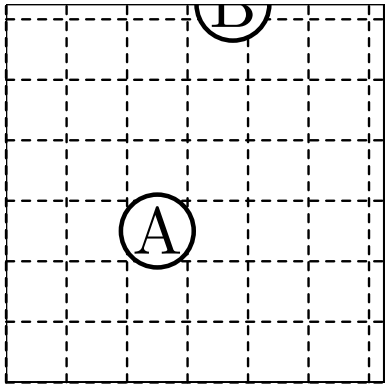
Wenn die Automatica-Bibiliothek verwendet werden soll, muss sie über den Befehl `input automat.mp` eingebunden werden. Bevor andere Befehle aus ihr verwendet werden können, muss `BeginAutomata;` aufgerufen werden, wodurch alle Parameter / Variablen initialisiert werden. Und am Ende darf der Aufruf `EndAutomata;` nicht fehlen, wodurch, wenn gewünscht, Rahmen (Frame) und Gitter (Grid) gezeichnet werden.

2.2 Rahmen

Wenn das Bild in seiner Größe beschränkt werden soll, kann man einen mit dem Befehl `SetFrame(`*llc*`,` *urc*`);` festlegen. Dazu müssen die linke untere Ecke *llc* und die rechte obere Ecke *urc* (vom Typ `pair`) angegeben werden. Dieser Rahmen wird nur dann angezeigt, wenn ein `ShowFrame;` hinzugefügt wird. Mit dem Befehl `HideFrame;` kann der Show-Befehl wieder rückgängig gemacht werden. Das letzte `ShowFrame;` bzw. `HideFrame;` zählt. Wenn der Rahmen angezeigt wird, kann man leicht sehen, wo man korrigieren muss, um im Bild zu bleiben. Der Rahmen kann aber auch dazu genutzt werden, alles, was außerhalb von ihm liegt, wegzuschneiden. Dazu muss `RestrictSize;` aufgerufen werden.

2.3 Gitter

Um das Zeichnen zu erleichtern, kann ein Gitter unter den Automaten gelegt werden. Mit der Funktion `SetGridDistance(`*Griddistance*`);` wird der Abstand zwischen zwei Gitterlinien (Default 1cm) auf den Wert *Griddistance* gesetzt. Angezeigt wird das Gitter mit dem Befehl `ShowGrid;` — entweder für das gesamte Bild oder, wenn ein Rahmen gesetzt ist, nur innerhalb des Rahmens. Vermutlich nicht notwendig, aber trotzdem möglich ist, das Gitter mit `HideGrid;` wieder zu deaktivieren. Das letzte `ShowGrid;` bzw. `HideGrid;` wird genommen.




```
BeginAutomata;
%%% Rahmen
SetFrame((0, 0), (5, 5));
ShowFrame;
RestrictSize;
%%% Gitter
SetGridDistance(0.8cm);
ShowGrid;
%%% Zustaende
State.A((2cm, 2cm), "A");
State.B((3cm, 5cm), "B");
EndAutomata;
```

3 Zustände

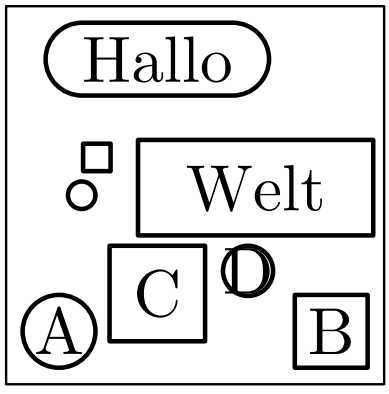
Ein endlicher Automat besteht aus Zuständen und Transitionen. Zuerst zu den Zuständen: Im Wesentlichen hat man bei Automatica die Wahl zwischen runden und rechteckigen sowie zwischen gewöhnlichen Zuständen und Start- und Endzuständen.

3.1 Normale Zustände

Einen runden Zustand fügt man über den Befehl `State.`*Name*`(`*(x, y)*`,` *Label*`);` ins Bild ein und zwar an die Position *(x, y)* mit dem angegebenen Label *Label*, das in den Kreis geschrieben wird. Der *Name* ist ausschliesslich für die Identifikation des Zustands gedacht. Er muss eindeutig sein und darf nur aus Buchstaben bestehen. Für einen quadratischen Zustand wird der Befehl `QState.`*Name*`(`*(x, y)*`,` *Label*`);` verwendet. Daneben gibt es die Möglichkeit, die Zustandsgröße an das Label anzupassen, indem ein sogenannter variabler Zustand gewählt wird. Die Befehle hierfür sind `StateVar.`*Name*`(`*(x, y)*`,` *Label*`);` und `QStateVar.`*Name*`(`*(x, y)*`,` *Label*`);`.

3.2 Zustandsgröße

Es gibt vier festgelegte Größen, nämlich „groß“ (Durchmesser 1.2cm), „mittel“ (0.9cm), „klein“ (0.6cm) und „sehr klein“ (0.3cm). Die Defaultgröße ist „mittel“. Die Größe kann durch die Befehle `LargeState;` („groß“), `MediumState;` („mittel“) und `SmallState;` („klein“) gesetzt werden. Sehr kleine Zustände kann man nur mit den Befehlen `VSState.`*Name*`(`*(x, y)*`);` und `VSQState.`*Name*`(`*(x, y)*`);` zeichnen. Die sehr kleinen Zustände sind zu klein für Label, deshalb können nur die Koordinaten *(x,y)* und der Name angegeben werden. Sollen andere Zustände ohne Label gezeichnet werden, so ist die Funktion einfach mit einem leeren String aufzurufen.

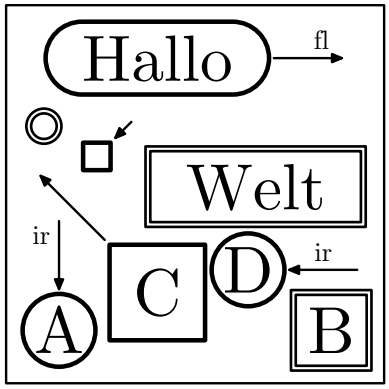



```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
%%% Zustaende
State.A((0.7cm, 0.7cm), "A");
QState.B((4.3cm, 0.7cm), "B");
LargeState;
QState.C((2cm, 1.2cm), "C");
SmallState;
State.D((3.2cm,1.5cm),"D");
VSState.E((1cm, 2.5cm));
MediumState;
VSQState.F((1.2cm,3cm));
StateVar.G((2cm, 4.3cm), "Hallo");
LargeState;
QStateVar.H((3.3cm, 2.6cm), "Welt");
EndAutomata;
```

Wenn ein Durchmesser ungleich den genannten gewünscht ist, dann gibt es noch die Möglichkeit, ihn mit `FixStateDiameter(`*Diam*`);` zu setzen, wobei *Diam* der Durchmesser ist.

3.3 Start- und Endzustände

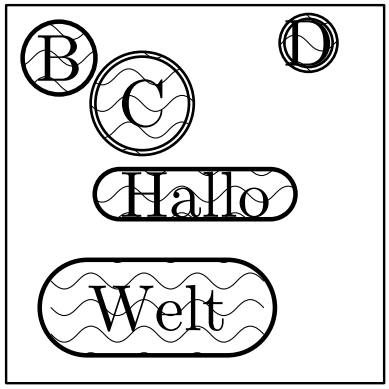
Runde Zustände mit doppelter Umrandung (Endzustände) können mit dem Befehl `FinalState.`*Name*`(`*(x, y)*`,` *Label*`);` an die Position *(x, y)* gezeichnet werden. Das Label wird wie gewohnt im Zustand platziert. Der Name dient allein der Identifizierung. Quadratische Zustände mit doppelte Umrandung erhält man mit `FinalQState.`*Name*`(`*(x, y)*`,` *Label*`);`. Alternativ kann das doppelte und einfache Umranden des Zustands auch mit den Befehlen `StateLineDouble;` und `StateLineSimple;` gesteuert werden, was den Vorteil hat, dass auch variable und sehr kleine Zustände so gezeichnet werden.
Start- und Endzustände können (auch) durch Pfeile gekennzeichnet werden, die entweder auf den Zustand oder von ihm weg zeigen. Die Standardpfeile ohne Label erhält man über `Initial.`*Name*`(`*Direction*`);` und `Final.`*Name*`(`*Direction*`);`.
Dabei muss der *Name* mit dem Zustandsnamen übereinstimmen, an den der Pfeil gezeichnet werden soll. *Direction* gibt die Richtung an; möglich sind nur `‘‘n’’` (Norden) für oben, `‘‘s’’` (Süden), `‘‘w’’` (Westen), `‘‘e’’` (Osten), `‘‘nw’’` (Nordwest), `‘‘ne’’` (Nordosten), `‘‘sw’’` (Südwesten) und `‘‘se’’` (Südosten).
Für Pfeile mit Label (links oder rechts) gibt es zusätzlich die Befehle:
`InitialL.`*Name*`(`*Direction*`,` *Position*`,` *Label*`);`
`InitialR.`*Name*`(`*Direction*`,` *Position*`,` *Label*`);`
`FinalL.`*Name*`(`*Direction*`,` *Position*`,` *Label*`);`
`FinalR.`*Name*`(`*Direction*`,` *Position*`,` *Label*`);`
Die *Position* des Labels *Label* wird dabei als Zahl zwischen 0 und 1 angegeben und durch Multiplikation mit der Länge des Pfeils bestimmt.




```
BeginAutomata;
SetFrame((0,0),(5cm,5cm)); ShowFrame;
%%% Zustaende
State.A((0.7cm, 0.7cm), "A");
FinalQState.B((4.3cm, 0.7cm), "B");
LargeState; QState.C((2cm,1.2cm),"C");
MediumState; State.D((3.2cm,1.5cm),"D");
StateLineDouble;
VSState.E((0.5cm, 3.4cm));
QStateVar.H((3.3cm, 2.6cm), "Welt");
StateLineSimple;
VSQState.F((1.2cm, 3cm));
StateVar.G((2cm, 4.3cm), "Hallo");
%%% Initial, Final
Final.C("nw"); InitialR.D("e", 0.5, "ir");
Initial.F("ne"); FinalL.G("e", 0.7, "fl");
InitialR.A("n", 0.2, "ir");
EndAutomata;
```

3.4 Zustände mit Hintergrundbild

Wie später bei den Parametern noch beschrieben wird, kann man in die Zustände vertikale, horizontale oder diagonale Linien zeichnen oder den Zustand ganz ausmalen lassen. Wem das nicht reicht, der kann ein eigenes Hintergrundbild kreieren und die Zustände mit dem Zusatz `WithPic` aufrufen. Für die runden Zustände sind das die Befehle:
`StateWithPic.`*Name*`(`*(x, y)*`,` *Label*`,` *Pic*`);`
`StateVarWithPic.`*Name*`(`*(x, y)*`,` *Label*`,` *Pic*`);`
`VSStateWithPic.`*Name*`(`*(x, y)*`,` *Pic*`);`
`FinalStateWithPic.`*Name*`(`*(x, y)*`,` *Label*`,` *Pic*`);`
Und für die rechteckigen Zustände:
`QStateWithPic.`*Name*`(`*(x, y)*`,` *Label*`,` *Pic*`);`
`QStateVarWithPic.`*Name*`(`*(x, y)*`,` *Label*`,` *Pic*`);`
`VSQStateWithPic.`*Name*`(`*(x, y)*`,` *Pic*`);`
`FinalQStateWithPic.`*Name*`(`*(x, y)*`,` *Label*`,` *Pic*`);`
Pic ist dabei das METAPOST-Bild (vom Typ picture).



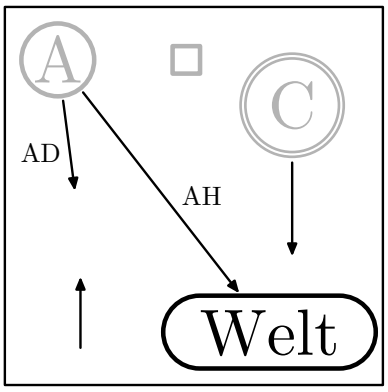

```
%%% Hintergrundbild (Metapost)
picture bsppic;
path p[];
p1=(0,0){curl 0}..(5pt,-3pt)..{curl 0}(10pt, 0);
p2=p1..(p1 yscaled-1 shifted(10pt,0));
p0=p2;
for i=1 upto 3:
  p0:=p0..p2 shifted (i*20pt,0); endfor
for j=0 upto 8:
  draw p0 shifted (0,j*10pt); endfor
bsppic := currentpicture;
currentpicture := nullpicture;
%%% Automat
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
StateWithPic.B((0.7cm,4.3cm),"B",bsppic);
LargeState;
FinalStateWithPic.C((1.8cm,3.7cm),"C",bsppic);
SmallState;
FinalStateWithPic.D((4cm,4.5cm),"D",bsppic);
StateVarWithPic.G((2.5cm,2.5cm),"Hallo",bsppic);
LargeState;
StateVarWithPic.H((2cm,1cm),"Welt",bsppic);
EndAutomata;
```

3.5 Gedimmte Zustände

Alle Zustände, die sich in einem DimState-Block befinden, d. h., zwischen `DimState;` und `RstState;`, werden in hellem Grau gezeichnet. Für die gedimmten Zustände gibt es eigene Parameter (s. Abschnitt „Parameter“).

3.6 Zustände ausblenden

Zustände, die nach einem `HideState;` stehen, werden ausgeblendet. Nichtsdestotrotz können Pfeile (Anfangs- / Endzustand), Loops und Transitionen an diese Zustände gezeichnet werden. Das `HideState;` kann durch ein `ShowState;` wieder aufgehoben werden.




```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
%%% Zustaende
DimState;
State.A((0.7cm, 4.3cm), "A");
VSQState.B((2.4cm, 4.3cm));
LargeState;
FinalState.C((3.8cm, 3.7cm), "C");
RstState;
MediumState;
HideState;
FinalQState.D((1cm, 2cm), "D");
ShowState;
StateVar.H((3.5cm, 0.7cm), "Welt");
%%% Initial, Final
Final.C("s");
Initial.D("s");
%%% Transitionen
EdgeL(A, H, "AH", 0.6);
EdgeR(A, D, "AD", 0.6);
EndAutomata;
```

4 Transitionen

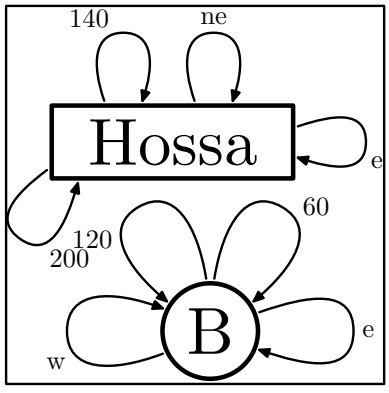
Eine Transition wird als Pfeil zwischen zwei Zuständen gezeichnet. Wenn der Pfeil zum selben Zustand zurückgeht, wird die Transition auch als Loop bezeichnet.

4.1 Loops in Standardrichtungen

Die Standardloops haben alle dieselbe Drehrichtung nach rechts. Zur Auswahl stehen Loops mit Öffnungswinkel 44 Grad und 60 Grad in den acht Himmelsrichtungen, `N` (Nord, oben), `NE` (Nordost, oben rechts), `E` (Ost, rechts), `SE` (Südost, unten rechts), `S` (Süd, unten), `SW` (Südwest, unten links), `W` (West, links) und `NW` (Nordwest, oben links). Loops mit Öffnungswinkel 60 Grad:
`LoopN(`*State*`,` *Label*`,` *Position*`);` `LoopNE(`*State*`,` *Label*`,` *Position*`);`
`LoopE(`*State*`,` *Label*`,` *Position*`);` `LoopSE(`*State*`,` *Label*`,` *Position*`);`
`LoopS(`*State*`,` *Label*`,` *Position*`);` `LoopSW(`*State*`,` *Label*`,` *Position*`);`
`LoopW(`*State*`,` *Label*`,` *Position*`);` `LoopNW(`*State*`,` *Label*`,` *Position*`);`
Loops mit Öffnungswinkel 44 Grad:
`CLoopN(`*State*`,` *Label*`,` *Position*`);` `CLoopNE(`*State*`,` *Label*`,` *Position*`);`
`CLoopE(`*State*`,` *Label*`,` *Position*`);` `CLoopSE(`*State*`,` *Label*`,` *Position*`);`
`CLoopS(`*State*`,` *Label*`,` *Position*`);` `CLoopSW(`*State*`,` *Label*`,` *Position*`);`
`CLoopW(`*State*`,` *Label*`,` *Position*`);` `CLoopNW(`*State*`,` *Label*`,` *Position*`);`
State ist der Zustand, der mit dem Loop versehen werden soll. Das *Label* wird vom Pfeilausgang ausgehend an die Parameterstelle *Position* gesetzt, die einen Wert zwischen 0 und 1 haben muss.

4.2 Loops in beliebigen Richtungen

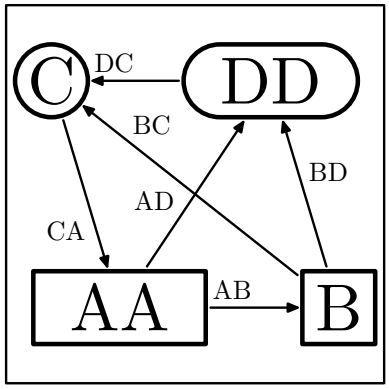
Wenn die Standardrichtungen `N`, `NE`, ..., `NW` nicht ausreichen und / oder wenn die Drehrichtung (`L`: links, `R`: rechts) wählbar sein soll, dann gibt es noch die Möglichkeit, auf einen der etwas umfangreicheren Befehle zurückzugreifen:
`LoopL(`*State*`,` *Direction*`,` *Label*`,` *Position*`);`
`LoopR(`*State*`,` *Direction*`,` *Label*`,` *Position*`);`
`CLoopL(`*State*`,` *Direction*`,` *Label*`,` *Position*`);`
`CLoopR(`*State*`,` *Direction*`,` *Label*`,` *Position*`);`
Dabei muss die Richtung *Direction* als Winkel in Grad angegeben werden.




```
BeginAutomata;
SetFrame((0,0),(5cm,5cm)); ShowFrame;
%%% Zustaende
QStateVar.A((2.2cm, 3.2cm), "Hossa");
LargeState;
State.B((2.7cm, 0.7cm), "B");
%%% Loops
LoopNE(A,"ne",0.5); CLoopE(A,"e",0.55);
LoopL(A, 140, "140", 0.45);
CLoopR(A, 200, "200", 0.57);
LoopW(B, "w", 0.4); CLoopE(B,"e", 0.5);
LoopL(B, 60, "60", 0.6);
LoopR(B, 120, "120", 0.7);
EndAutomata;
```

4.3 Gerade Transitionen

Gerade Transitionen mit Label links bzw. rechts vom Pfeil werden mit den Makros `EdgeL(`*StateA*`,` *StateB*`,` *Label*`,` *Position*`);` und `EdgeR(`*StateA*`,` *StateB*`,` *Label*`,` *Position*`);` erzeugt. Dabei wird der Pfeil vom Zustand *StateA* zum Zustand *StateB* gezeichnet. Das *Label* wird an die Parameterstelle *Position* gesetzt, die im Bereich zwischen 0 und 1 liegen muss.

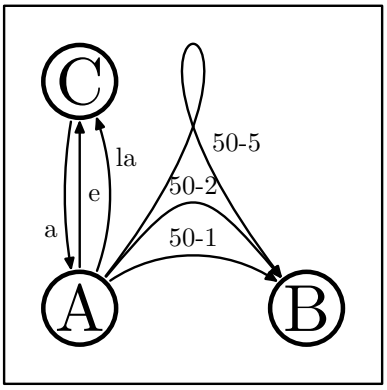



```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
%%% Zustaende
QStateVar.A((1.5cm, 1cm), "AA");
QState.B((4.4cm, 1cm), "B");
State.C((0.6cm, 4cm), "C");
StateVar.D((3.5cm, 4cm), "DD");
%%% Transitionen
EdgeL(A, B, "AB", 0.25);
EdgeL(A, D, "AD", 0.35);
EdgeR(B, C, "BC", 0.8);
EdgeR(B, D, "BD", 0.55);
EdgeR(C, A, "CA", 0.65);
EdgeR(D, C, "DC", 0.75);
EndAutomata;
```

4.4 Bogen

Es gibt drei verschiedene Möglichkeiten, Bogen zu zeichnen, die bzgl. der geraden Linie zwischen zwei Zuständen in einem Winkel α (15 Grad, 30 Grad oder beliebig) den Zustand *StateA* verlassen und wieder bzgl. der geraden Linie auf der selben Seite im Winkel α in den Zustand *StateB* einmünden. Zusätzlich zum Winkel kann der Bogen rechts oder links von der gedachten geraden Verbindungslinie gezeichnen werden. Das Label wird bei Bogen auf der rechten bzw. linken Seite auch rechts oder links platziert.
Bogen mit Winkel 15 Grad:
`ArcL(`*StateA*`,` *StateB*`,` *Label*`,` *Position*`);`
`ArcR(`*StateA*`,` *StateB*`,` *Label*`,` *Position*`);`
Bogen mit Winkel 30 Grad:
`LArcL(`*StateA*`,` *StateB*`,` *Label*`,` *Position*`);`
`LArcR(`*StateA*`,` *StateB*`,` *Label*`,` *Position*`);`
Bogen mit variablem Winkel:
`VArcL(`*StateA*`,` *StateB*`,` *Angle*`,` *Eccentricity*`,` *Label*`,` *Position*`);`
`VArcR(`*StateA*`,` *StateB*`,` *Angle*`,` *Eccentricity*`,` *Label*`,` *Position*`);`
Bei den Bogen mit variablen Winkeln kann man zusätzlich die Exzentrizität *Eccentricity* wählen. Kurven in METAPOST sind Bezier-Kurven, die mit Hilfe von vier Kontrollpunkten konstruiert werden. Die Exzentrizität ist ein Koeffizient, mit dem die Abstände zwischen erstem und zweitem Kontrollpunkt und

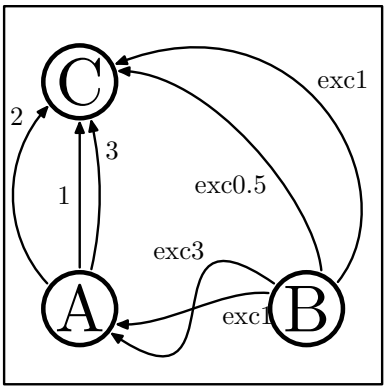
zwischen viertem und drittem Kontrollpunkt gestreckt bzw. gestaucht werden können.




```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
%%% Zustaende
State.A((1cm, 1cm), "A");
State.B((4cm, 1cm), "B");
State.C((1cm, 4cm), "C");
%%% Transitionen
EdgeR(A, C, "e", 0.5);
ArcR(C, A, "a", 0.75);
LArcR(A, C, "la", 0.75);
VArcL(A, B, 50, 1, "50-1", 0.5);
VArcL(A, B, 50, 2, "50-2", 0.5);
VArcL(A, B, 50, 5, "50-5", 0.85);
EndAutomata;
```

4.5 Kurven

Die sogenannten „Kurven“ sind eine Erweiterung der Bogen mit variablem Winkel (s. vorherigen Abschnitt). Neben der Exzentrizität lassen sich jetzt auch Ausgangwinkel und Eingangswinkel getrennt angeben. Dabei ist zu beachten, dass die Winkel nicht mehr von der geraden Verbindungslinie zwischen den Zuständen gemessen werden, sondern von der Horizontalen Richtung Osten aus.
`VCurveL(`*StateA*, *StateB*, *AngleA*, *AngleB*, *Eccentricity*, *Label*, *Position*`);`
`VCurveR(`*StateA*, *StateB*, *AngleA*, *AngleB*, *Eccentricity*, *Label*, *Position*`);`



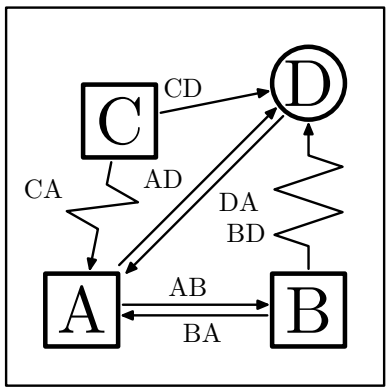

```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
%%% Zustaende
State.A((1cm, 1cm), "A");
State.B((4cm, 1cm), "B");
State.C((1cm, 4cm), "C");
%%% Transitionen
VCurveL(A, C, 90, -90, 1, "1", 0.5);
VCurveL(A, C, 151, -151, 1, "2", 0.85);
VCurveR(A, C, 70, -70, 1, "3", 0.8);
VCurveL(B, A, 140, -40, 1, "exc1", 0.35);
VCurveR(B, A, 140, -40, 3, "exc3", 0.25);
VCurveL(B, C, 30, 30, 0.5, "exc0.5", 0.4);
VCurveR(B, C, 30, 30, 1, "exc1", 0.55);
EndAutomata;
```

4.6 Zickzack-Transitionen

Alternativ zu geraden Transitionen und Kurven gibt es die Möglichkeit, eine Zickzack-Kurve als Transition zwischen den Zuständen *StateA* und *StateB* zu wählen:
`ZZEdgeL(`*StateA*`,` *StateB*`,` *Label*`,` *Position*`);`
`ZZEdgeR(`*StateA*`,` *StateB*`,` *Label*`,` *Position*`);`

4.7 Entgegengesetzte geradlinige Transitionen

Um entgegengesetzte geradlinige Transitionen zwischen zwei Zuständen nebeneinander darstellen zu können, gibt es den Befehl `ForthBackOffset;`, der bewirkt, dass sie leicht nach links verschoben gezeichnet werden. Mit `RstEdgeOffset;` wird dieser Modus wieder ausgeschaltet.

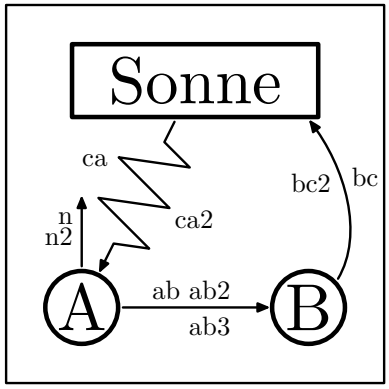



```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
%%% Zustaende
QState.A((1cm, 1cm), "A");
QState.B((4cm, 1cm), "B");
QState.C((1.5cm, 3.5cm), "C");
State.D((4cm, 4cm), "D");
%%% Transitionen
ForthBackOffset;
EdgeL(A, D, "AD", 0.45);
EdgeL(D, A, "DA", 0.45);
EdgeL(A, B, "AB", 0.45);
EdgeL(B, A, "BA", 0.45);
RstEdgeOffset;
EdgeL(C, D, "CD", 0.45);
ZZEdgeR(C, A, "CA", 0.25);
ZZEdgeL(B, D, "BD", 0.25);
EndAutomata;
```

5 Verschiedenes

5.1 Zusatzlabel

An jede Transition, jeden Loop und jeden Pfeil eines Angangs- oder Endzustands können weitere Label mit den Befehlen `LabelL(`*Position*`, `*Label*`);` und `LabelR(`*Position*`, `*Label*`);` geschrieben werden. `L` und `R` stehen dabei für links und rechts (vom Pfeil). Wie gewohnt ist *Position* eine Zahl zwischen 0 und 1, die den Punkt der Kurve bestimmt, an den das *Label* gezeichnet werden soll.



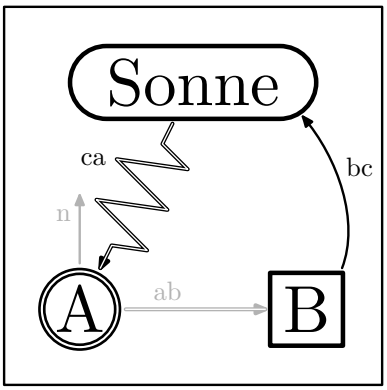

```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
%%% Zustaende
State.A((1cm, 1cm), "A");
State.B((4cm, 1cm), "B");
QStateVar.C((2.5cm, 4cm), "Sonne");
%%% Initial, Final
FinalL.A("n", 0.7, "n");
LabelL(0.44, "n2");
%%% Transitionen
EdgeL(A, B, "ab", 0.3);
LabelL(0.6, "ab2");
LabelR(0.6, "ab3");
ZZEdgeR(C, A, "ca", 0.5);
LabelL(0.4, "ca2");
VCurveR(B, C, 30, 30, 0.7, "bc", 0.55);
LabelL(0.7, "bc2");
EndAutomata;
```

5.2 Gedimmte Transitionen

Analog zu gedimmten Zuständen, werden alle Transitionen (inkl. Loops und Pfeile bei Anfangs- und Endzuständen), die sich in einem `DimEdge`-Block, also zwischen einem `DimEdge;` und einem `RstEdge;` befinden, in hellem Grau gezeichnet.

5.3 Transitionen mit doppelter Linie

Pfeile (Anfangs- und Endzustände), Loops und Transitionen können mit Doppelstrich gezeichnet werden. Um diesen Modus anzuschalten, verwende man `EdgeLineDouble;` und zum Ausschalten `EdgeLineSimple;`.



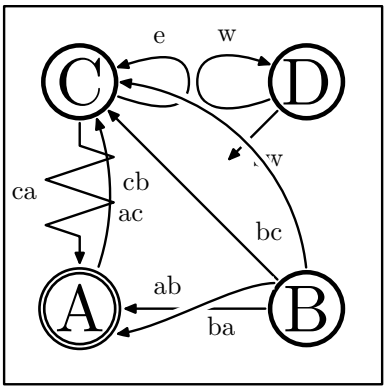

```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm)); ShowFrame;
%%% Zustaende
FinalState.A((1cm, 1cm), "A");
QState.B((4cm, 1cm), "B");
StateVar.C((2.5cm, 4cm), "Sonne");
%%% Initial, Final, Transitionen
DimEdge;
FinalL.A("n", 0.7, "n");
EdgeLineDouble;
EdgeL(A, B, "ab", 0.3);
RstEdge;
ZZEdgeR(C, A, "ca", 0.5);
EdgeLineSimple;
VCurveR(B, C, 30, 30, 0.7, "bc", 0.55);
EndAutomata;
```

5.4 Transitionen mit Rändern

Anfangs- und Endpfeile, Loops und Transitionen können mit Rändern versehen werden, was im Wesentlichen dafür gedacht ist, dass zwei sich überschneidende Pfeile voneinander getrennt werden. Alle Pfeile, Loops, Transitionen, die sich in einem `EdgeBorder`-Block befinden, also zwischen einem `EdgeBorder;` und einem `EdgeBorderOff;` werden mit einem Rand versehen (Default-Farbe Weiß).

5.5 Pfeilrichtung umdrehen

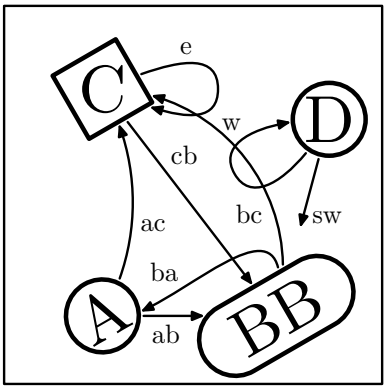
Alle Pfeile, Loops und Transitionen, die sich in einem `ReverseArrow`-Block, also zwischen einem `ReverseArrow;` und einem `StraightArrow;` befinden, werden umgedreht, d. h., die Pfeilspitze wird an das andere Ende gezeichnet. Das bietet sich u. a. für Standard-Loops an, die sich sonst nur rechtsherum drehen.




```
BeginAutomata;
SetFrame((0,0),(5cm,5cm)); ShowFrame;
FinalState.A((1cm, 1cm), "A");
State.B((4cm, 1cm), "B");
State.C((1cm,4cm),"C"); State.D((4cm,4cm),"D");
%%% Initial, Final, Transitionen
EdgeBorder;
FinalL.D("sw", 0.7, "sw"); LoopW(D, "w", 0.8);
ReverseArrow; CLoopE(C, "e", 0.2);
EdgeL(A, B, "ab", 0.3); EdgeR(C, B, "cb", 0.3);
StraightArrow;
VArcL(B, C, -50, 1, "bc", 0.2);
VCurveL(B, A, 100, -80, 0.7, "ba", 0.5);
EdgeBorderOff;
LArcR(A, C, "ac",0.35); ZZEdgeR(C, A,"ca",0.5);
EndAutomata;
```

5.6 Translation und Rotation

Um Zustände, Loops und Pfeile von Anfangs- oder Endzuständen zu verschieben und / oder zu drehen, kann der TraRot-Befehl benutzt werden. Alle solchen Objekte, die sich zwischen `TraRot(`*Translation*`, `*Rotation*`);` und `TraRotEnd;` befinden, werden um *Translation* (vom Typ `pair`) verschoben und um den Mittelpunkt des Zustandes im Winkel *Rotation* (vom Typ `numeric`) gedreht. Dabei werden die Label der Loops und Pfeile nur verschoben, nicht aber rotiert. Dasselbe gilt für die Label der Zustände in der Default-Einstellung. Mit dem Befehl `SwivelLabel;` kann aber dafür gesorgt werden, dass die Zustandslabel gedreht dargestellt werden. Mit `RigidLabel;` erhält man wieder die Dafault-Einstellung. Normale Transitionen werden nach wie vor (bei Kurven mit den angegebenen Winkeln!) von Zustand zu Zustand gezeichnet, ob sie im TraRot-Block sind oder nicht.




```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
State.D((4.3cm, 3.5cm), "D");
%%% TraRot-Block
TraRot((-0.7cm, 0.4cm), 30);
SwivelLabel;
State.A((2cm, 0.5cm), "A");
StateVar.B((4.3cm, 0.5cm), "BB");
RigidLabel;
QState.C((2cm, 3.5cm), "C");
FinalL.D("sw", 0.7, "sw");
LoopW(D, "w", 0.8);
EdgeR(A, B, "ab", 0.38);
VCurveR(B, A, 80, -30, 1.2, "ba", 0.75);
TraRotEnd;
%%% TraRot-Block-Ende
CLoopE(C, "e", 0.2);
EdgeL(C, B, "cb", 0.3);
VArcL(B, C, -53, 1, "bc", 0.3);
LArcR(A, C, "ac", 0.35);
EndAutomata;
```

6 Parameter

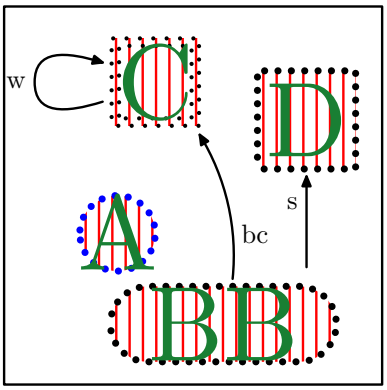
Alle in diesem Abschnitt aufgelisteten Parameter haben einen Default-Wert und einen veränderlichen Wert. Diese Werte lassen sich jeweils durch drei Makros ändern. Mit `ChgParametername(`*Value*`);` wird der veränderliche Wert auf *Value* gesetzt, mit `SetParametername(`*Value*`);` wird der Default-Wert auf *Value* gesetzt, und mit `RstParametername;` wird der veränderliche Wert auf Default-Wert gesetzt.

6.1 Zustände

Die folgenden Parameter können mit den oben genannten Makros verändert werden:

Name	Typ	Default-Wert
`StateLineColor`	`color`	`black`
`StateLabelColor`	`color`	`black`
`StateFillColor`	`color`	`white`
`StateLineStyle`	eigene Namen	`solid`
`StateLineWidth`	`numeric`	`1.8pt`
`StateLabelScale`	`numeric`	`1.0`
`StateFillStatus`	eigene Namen	`solid`

Die beiden letzten Parameter beziehen sich auch auf gedimmte Zustände. Bei den Stilen (Styles) kann zwischen `solid` („ausgefüllt“, default) `dash` („gestrichelt“) `dot` („gepunktet“) und `none` („keine Linie“) gewechselt werden. Der Parameter `StateLabelScale` ist ein Koeffizient, um die Größe des Labels zu ändern, der Parameter `StateFillStatus` gibt an, wie der Hintergrund des Zustands gemalt werden soll (sofern kein Hintergrundbild angegeben ist). Die Möglichkeiten sind: `solid` („ausgefüllt“), `vlines` („vertikale Linien“) `hlines` („horizontale Linien“), `crosshatch` („diagonale Linien“) und `none` („kein Bild“, default).



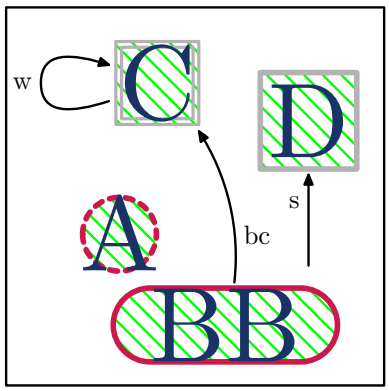

```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
%%% Zustaende und Parameter
ChgStateLineColor(blue);
SetStateLabelColor((0.1,0.5,0.2));
ChgStateFillColor(red);
ChgStateLineStyle(dot);
SetStateLineWidth(2.8pt);
ChgStateLabelScale(1.6);
ChgStateFillStatus(vlines);
State.A((1.5cm, 2cm), "A");
RstStateLineColor;
RstStateLabelColor;
StateVar.B((2.9cm, 0.8cm), "BB");
FinalQState.C((2cm, 4cm), "C");
LargeState;
QState.D((4cm, 3.5cm), "D");
%%% Initial, Final, Loops, Transitionen
InitialL.D("s", 0.7, "s");
CLoopW(C, "w", 0.5);
VArcR(B, C, 33, 1, "bc", 0.3);
EndAutomata;
```

6.2 Gedimmte Zustände

Für gedimmte Zustände gibt es die folgenden Parameter:

Name	Typ	Default-Wert
`DimStateLineStyle`	eigene Namen	`solid`
`DimStateLineColor`	`color`	`(0.7, 0.7, 0.7)`
`DimStateLineCoef`	`numeric`	`1.0`
`DimStateLabelColor`	`color`	`(0.7, 0.7, 0.7)`
`DimStateFillColor`	`color`	`white`

Die Breite der Umrandung wird bestimmt durch die Breite für normale Zustände multipliziert mit dem Koeffizienten `DimStateLineCoef`. Alle Parameter auf einmal lassen sich mit der Funktion `FixDimState(`*Linestyle*`,` *Linecolor*`,` *Linecoef*`,` *Labelcolor*`,` *Fillcolor*`);` setzen.




```
BeginAutomata;
SetFrame((0,0),(5cm,5cm)); ShowFrame;
%%% Zustaende und Parameter
DimState;
ChgDimStateLineColor((0.8,0.1,0.3));
ChgDimStateLabelColor((0.1,0.2,0.4));
SetDimStateFillColor(green);
ChgDimStateLineStyle(dash);
SetDimStateLineCoef(1.2);
ChgStateLabelScale(1.6);
ChgStateFillStatus(crosshatch);
State.A((1.5cm, 2cm), "A");
RstDimStateLineStyle;
StateVar.B((2.9cm, 0.8cm), "BB");
RstDimStateLineColor;
FinalQState.C((2cm, 4cm), "C");
LargeState; QState.D((4cm,3.5cm),"D");
RstState;
%%% Initial, Loops, Transitionen
InitialL.D("s", 0.7, "s");
CLoopW(C, "w", 0.5);
VArcR(B, C, 33, 1, "bc", 0.3);
EndAutomata;
```

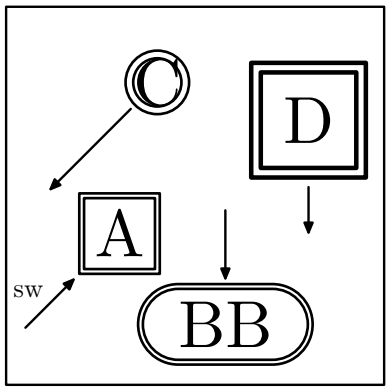
6.3 Zustände mit doppelter Umrandung

Die Parameter `StateLineDblCoef` und `StateLineDblSep` werden bei Zuständen mit doppelter Umrandung verwendet, um die Strichdicke der Umrandungen und den Abstand zwischen ihnen zu setzten. Es handelt sich hierbei um Koeffizienten, die mit der Breite des normalen Zustands (mit einer Umrandung) multipliziert

werden. Umrandungen und Abstand zusammen haben letztendlich eine Breite von: `StateLineWidth` * (2 * `StateLineDblCoef` + `StateLineDblSep`).

6.4 Anfangs- und Endzustände

Die Länge der Pfeile bei Anfangs- und Endzuständen entspricht dem Durchmesser des jeweiligen Zustandes. Für die einzelnen Größen „groß“, „mittel“, „klein“ und „sehr klein“ kann sie aber durch Setzen der folgenden Parameter verändert werden:
`ArrowOnLargeState`
`ArrowOnMediumState`
`ArrowOnSmallState`
`ArrowOnVerySmallState`
Diese Parameter sind Koeffizienten (Default-Wert = 1), die zur Default-Länge, also zum Zustandsdurchmesser, multipliziert werden.

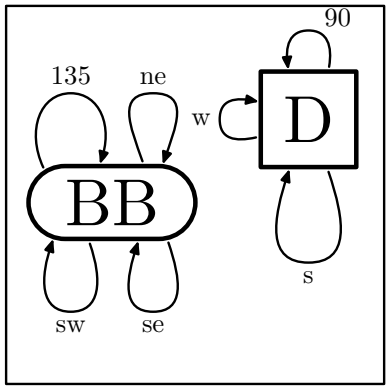



```
BeginAutomata;
SetFrame((0,0),(5cm,5cm)); ShowFrame;
%%% Zustaende und Parameter
ChgStateLineDblCoef(1.);
ChgStateLineDblSep(1.);
ChgArrowOnLargeState(0.5);
SetArrowOnSmallState(2.5);
StateLineDouble;
QState.A((1.5cm, 2cm), "A");
StateVar.B((2.9cm, 0.8cm), "BB");
SmallState; RstStateLineDblSep;
FinalState.C((2cm, 4cm), "C");
RstStateLineDblCoef;
LargeState;
QState.D((4cm, 3.5cm), "D");
StateLineSimple;
%%% Initial, Final
InitialL.A("sw", 0.5, "sw");
Initial.B("n");
Final.C("sw"); Final.D("s");
EndAutomata;
```

6.5 Loops

Die Größe der Loops entspricht bei kreisrunden Zuständen seinem Durchmesser (bei quadratischen und variablen Zuständen seiner Höhe), d. h., der Abstand zwischen Kreismittelpunkt und äußerstem Punkt des Loops beträgt 3/2 Zustandsdurchmesser. Für die einzelnen Größen „groß“, „mittel“, „klein“ und „sehr klein“ kann dieser Abstand durch Setzen der folgenden Parameter verändert werden.
`LoopOnLargeState`

`LoopOnMediumState`
`LoopOnSmallState`
`LoopOnVerySmallState`
Diese Parameter sind Koeffizienten (Default-Wert = 1), die zur Default-Länge, also zum Zustandsdurchmesser, multipliziert werden.
Die Öffnungswinkel der Loops sind 60 Grad (Loop) und 44 Grad (CLoop). Um diese Winkel zu ändern, müssen die Parameter `LoopAngle` (Default: 60) und `CLoopAngle` (Default: 44) neu gesetzt werden.




```
BeginAutomata;
SetFrame((0,0),(5cm,5cm)); ShowFrame;
%%% Zustaende
StateVar.B((1.4cm, 2.4cm), "BB");
LargeState;
QState.D((4cm, 3.5cm), "D");
%%% Loops und Parameter
ChgLoopOnLargeState(0.4);
ChgCLoopAngle(10);
ChgLoopAngle(100);
CLoopNE(B, "ne", 0.5);
LoopL(B, 135, "135", 0.5);
RstCLoopAngle;
RstLoopAngle;
CLoopSE(B, "se", 0.5);
LoopSW(B,"sw",0.5); CLoopW(D,"w",0.5);
LoopR(D, 90, "90", 0.4);
RstLoopOnLargeState;
CLoopS(D, "s", 0.5);
EndAutomata;
```

6.6 Transitionen allgemein

Die folgenden Parameter verändern das Erscheinungsbild der Transitionen:

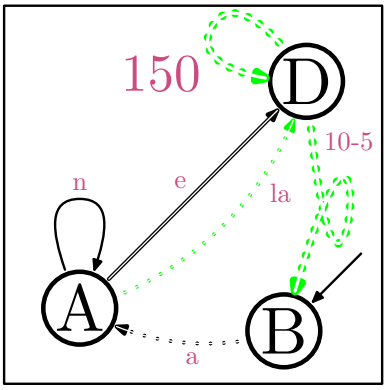
Name	Typ	Default-Wert
`EdgeLineColor`	`color`	`black`
`EdgeLabelColor`	`color`	`black`
`EdgeLineStyle`	eigene Namen	`solid`
`EdgeLineWidth`	`numeric`	`1pt`
`EdgeLabelScale`	`numeric`	`1`
`EdgeLineDblStatus`	`boolean`	`false`

Anders als der Name vermuten lassen könnte, beziehen sich diese Parameter nicht nur auf gerade Transitionen (Edges), sondern auch auf Bogen (Arcs), Kurven (Curves) sowie Loops und die Pfeile der Anfangs- und Endzustände. Für gedimmte Transitionen gibt es separate Parameter mit Ausnahme der letzten

beiden Parameter, die auch für gedimmte Transitionen gelten.
Bei den Stilen (Styles) kann zwischen `solid` („ausgefüllt“) `dash` („gestrichelt“) `dot` („gepunktet“) und `none` („keine Linie“) gewechselt werden. Der Parameter `EdgeLabelScale` ist ein Koeffizient, um die Größe des Labels zu ändern, der Parameter `EdgeLineDblStatus` stellt eine weitere Möglichkeit dar, zwischen einfachen und doppelten Linien zu wechseln.

6.7 Bogen

Die Standardwinkel für Bogen sind 15 Grad (`Arc`) und 30 Grad (`LArc`). Diese Werte können durch Neusetzen der Parameter `ArcAngle` und `LArcAngle` geändert werden.




```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
%%% Zustaende
State.A((1cm, 1cm), "A");
State.B((3.7cm, 0.7cm), "B");
State.D((4cm, 4cm), "D");
%%% Transitionen und Parameter
Initial.B("ne");
ChgEdgeLabelColor((0.8, 0.3, 0.5));
CLoopN(A, "n", 0.5);
ChgEdgeLineDblStatus(true);
EdgeL(A, D, "e", 0.5);
ChgEdgeLineStyle(dot);
ChgArcAngle(25);
ArcL(B, A, "a", 0.4);
ChgEdgeLineColor(green);
ChgLArcAngle(35);
LArcR(A, D, "la", 0.75);
ChgEdgeLineWidth(2);
VArcL(D, B, 10, 5, "10-5", 0.02);
ChgEdgeLabelScale(2);
LoopR(D, 150, "150", 0.65);
EndAutomata;
```

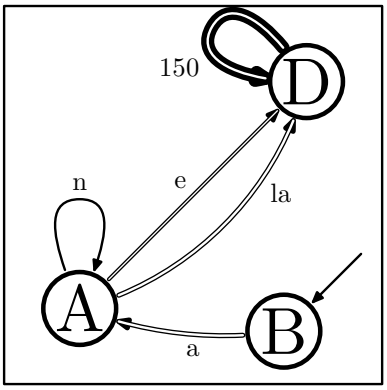
6.8 Ränder der Transitionen

Die Ränder der Transitionen werden dadurch erzeugt, dass erst eine breitere Linie in der Rand-Farbe gezogen wird, dann eine schmalere in der Transition-Farbe. Die Breite der ersten Linie kann verändert werden; sie ergibt sich aus dem Produkt der Transition-Linienbreite und dem Koeffizienten `EdgeLineBorderCoef`. Dieser sollte größer als 1 sein, damit der Rand zu sehen ist. Default-Wert ist 2,

d. h., der Rand auf jeder Seite beträgt die halbe Transition-Linienbreite. Die Farbe des Randes kann über den Parameter `EdgeLineBorderColor` gesetzt werden ist (Default: Weiß).

6.9 Transitionen mit Doppellinien

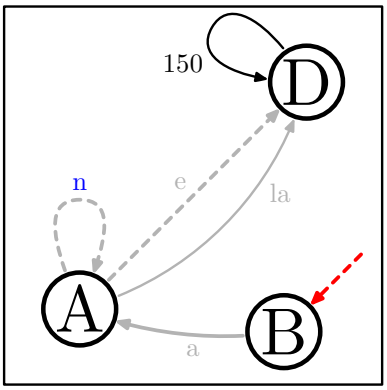
Zu Transitionen mit Doppellinien gibt es zwei Koeffizienten, die jeweils mit der Linienbreite normaler Transitionen (mit nur einer Linie) multipliziert werden. Der erste Koeffizient, `EdgeLineDblCoef`, ist für die Breite der Linien (Default: 0.5), der zweite Koeffizient, `EdgeLineDblSep` für die Breite des Raums zwischen den Linien (Default: 0.6).




```
BeginAutomata;
SetFrame((0, 0), (5cm, 5cm));
ShowFrame;
%%% Zustaende
State.A((1cm, 1cm), "A");
State.B((3.7cm, 0.7cm), "B");
State.D((4cm, 4cm), "D");
%%% Transitionen und Parameter
Initial.B("ne");
ChgEdgeLineBorderColor((0.8, 0.3, 0.5));
CLoopN(A, "n", 0.5);
ChgEdgeLineDblStatus(true);
EdgeL(A, D, "e", 0.5);
ChgEdgeLineDblSep(1);
ArcL(B, A, "a", 0.4);
ChgEdgeLineBorderCoef(3);
LArcR(A, D, "la", 0.75);
ChgEdgeLineDblCoef(2);
LoopR(D, 150, "150", 0.65);
EndAutomata;
```

6.10 Gedimmte Transitionen

Die Parameter für gedimmte Transitionen sind:
`DimEdgeLineColor`
`DimEdgeLabelColor`
`DimEdgeLineStyle`
`DimEdgeLineCoef`
Die Breite der gedimmten Transitionen ergibt sich aus der Breite normaler Transitionen, multipliziert mit dem Koeffizienten `DimEdgeLineCoef`. Bei den Stilen (Styles) kann man wechseln zwischen `solid` („ausgefüllt“, default) `dash` („gestrichelt“) `dot` („gepunktet“) und `none` („keine Linie“). Alle Parameter auf einmal können mit der Funktion `FixDimEdge(`*Linestyle*`,` *Linecolor*`,` *Linecoef*`,` *Labelcolor*`);` gesetzt werden.



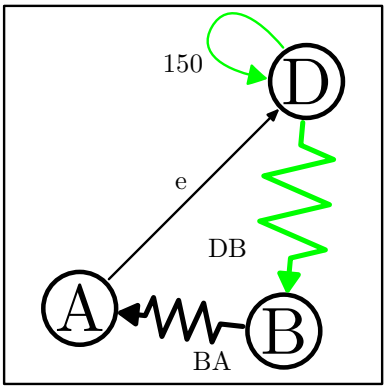

```
BeginAutomata;
SetFrame((0,0),(5cm,5cm)); ShowFrame;
State.A((1cm, 1cm), "A");
State.B((3.7cm, 0.7cm), "B");
State.D((4cm, 4cm), "D");
DimEdge; FixDimEdge(dash,red,1.5,blue);
Initial.B("ne");
RstDimEdgeLineColor;
CLoopN(A, "n", 0.5);
RstDimEdgeLabelColor;
EdgeL(A, D, "e", 0.5);
RstDimEdgeLineStyle;
ArcL(B, A, "a", 0.4);
RstDimEdgeLineCoef;
LArcR(A, D, "la", 0.75); RstEdge;
LoopR(D, 150, "150", 0.65);
EndAutomata;
```

6.11 Zickzack-Transition

Bei den Zickzack-Transitionen kann die Linienbreite mit `ZZLineWidth` (Koeffizient, der auf die normale Transitionenbreite multipliziert wird; Default: 1) und die Gesamtbreite, die die Zickzacklinie einnimmt, mit `ZZSize` (Default: 0.9cm) festgelegt werden. Andere relevante Einstellungen (Farbe etc.) werden durch Parameter für normale / gedimmte Transitionen mit gesetzt.

6.12 Pfeilspitzen

Die Pfeilköpfe (für alle Pfeile) lassen sich manipulieren über die Parameter `EdgeArrowAngle`, den Winkel zwischen den von der Pfeilspitze weggehenden Kanten des Pfeilkopfes, und `EdgeArrowLength`, die Länge dieser Kanten. Default-Werte sind 45 Grad und 4.




```
BeginAutomata;
SetFrame((0,0),(5cm,5cm)); ShowFrame;
State.A((1cm, 1cm), "A");
State.B((3.7cm, 0.7cm), "B");
State.D((4cm, 4cm), "D");
EdgeL(A, D, "e", 0.5);
SetEdgeArrowLength(7);
ChgEdgeArrowAngle(60);
ChgEdgeLineColor(green);
LoopR(D, 150, "150", 0.65);
ChgZZLineWidth(3);
ZZEdgeR(D, B, "DB", 0.75);
RstEdgeLineColor; ChgZZSize(0.5cm);
ZZEdgeL(B, A, "BA", 0.25);
EndAutomata;
```

A Übersicht über die Makros

Tabelle 1. Zeichenumgebung

Name	Argumente	Erklärung
`BeginAutomata`		Initialisierung der Parameter
`EndAutomata`		Einfügen von Gitter und Rahmen
`SetFrame`	pair, pair	Rahmen setzen
`ShowFrame`		Rahmen anzeigen
`HideFrame`		Rahmen verbergen
`RestrictSize`		Bildgröße durch Rahmen beschränken
`SetGridDistance`	numeric	Gitterabstand setzen
`ShowGrid`		Gitter anzeigen
`HideGrid`		Gitter verbergen

Tabelle 2. Designbefehle für Zustände

Name	Argumente	Erklärung
`HideState`		Zustände ausblenden
`ShowState`		Zustände anzeigen
`DimState`		Zustände dimmen
`RstState`		Zustände ungedimmt
`FixDimState`	numeric, color, numeric, color, color	Parameter für gedimmte Zustände setzen
`LargeState`		Zustandsdurchmesser groß
`MediumState`		Zustandsdurchmesser mittel
`SmallState`		Zustandsdurchmesser klein
`FixStateDiameter`	numeric	Zustandsdurchmesser setzen
`StateLineDouble`		Zustände mit doppelter Umrandung
`StateLineSimple`		Zustände mit einfacher Umrandung

Tabelle 3. Zeichenbefehle für Zustände

Name	Argumente	Erklärung
`State`	suffix, pair, string	Kreis mit Bezeichner zeichnen
`QState`	suffix, pair, string	Quadrat mit Bezeichner zeichnen
`StateVar`	suffix, pair, string	an den Bezeichner angepasstes Oval zeichnen
`QStateVar`	suffix, pair, string	an den Bezeichner angepasstes Rechteck zeichnen
`VSState`	suffix, pair	kleinen Kreis zeichnen
`VSQState`	suffix, pair	kleines Quadrat zeichnen
`FinalState`	suffix, pair, string	Kreis mit doppelter Umrandung
`FinalQState`	suffix, pair, string	Quadrat mit doppelter Umrandung
`StateWithPic`	suffix, pair, string, picture	Kreis mit Bild und Bezeichner
`QStateWithPic`	suffix, pair, string, picture	Qudrat mit Bild und Bezeichner
`StateVarWithPic`	suffix, pair, string, picture	Oval mit Bild und Bezeichner
`QStateVarWithPic`	suffix, pair, string, picture	Rechteck mit Bild und Bezeichner
`VSStateWithPic`	suffix, pair, string, picture	kleiner Kreis mit Bild
`VSQStateWithPic`	suffix, pair, string, picture	kleines Quadrat mit Bild
`FinalStateWithPic`	suffix, pair, string, picture	Kreis mit doppelter Umrandung und Bild
`FinalQStateWithPic`	suffix, pair, string, picture	Quadrat mit doppelter Umrandung und Bild

Tabelle 4. Zeichenbefehle für Start- und Endpfeile

Name	Argumente	Erklärung
`Initial`	suffix, numeric	eingehenden Pfeil an Zustand zeichnen
`Final`	suffix, numeric	wegzeigenden Pfeil an Zustand zeichnen
`InitialL`	suffix, numeric, numeric, string	eingehender Pfeil mit Label links
`InitialR`	suffix, numeric, numeric, string	eingehender Pfeil mit Label rechts
`FinalL`	suffix, numeric, numeric, string	wegzeigender Pfeil mit Label links
`FinalR`	suffix, numeric, numeric, string	wegzeigender Pfeil mit Label rechts

Tabelle 5. Designbefehle für Transitionen

Name	Argumente	Erklärung
`DimEdge`		Transitionen dimmen
`RstEdge`		Transitionen ungedimmt
`FixDimEdge`	numeric, color, numeric, color	Parameter für gedimmte Transitionen setzen
`EdgeLineDouble`		Transitionen mit Doppellinie
`EdgeLineSimple`		Transitionen mit einfacher Linie
`EdgeBorder`		Transitionen mit Rand
`EdgeBorderOff`		Transitionen ohne Rand
`ForthBackOffset`		gerade Transitionen nach links verschieben
`RstEdgeOffset`		gerade Transitionen normal zeichnen
`ReverseArrow`		Pfeile umdrehen
`StraightArrow`		Pfeile nicht umdrehen

Tabelle 6. Zeichenbefehle für Transitionen 1 (Loops)

Name	Argumente	Erklärung
`LoopN`	suffix, string, numeric	60 Grad Loop oben
`LoopNE`	suffix, string, numeric	60 Grad Loop rechts oben
`LoopE`	suffix, string, numeric	60 Grad Loop rechts
`LoopSE`	suffix, string, numeric	60 Grad Loop rechts unten
`LoopS`	suffix, string, numeric	60 Grad Loop unten
`LoopSW`	suffix, string, numeric	60 Grad Loop links unten
`LoopW`	suffix, string, numeric	60 Grad Loop links
`LoopNW`	suffix, string, numeric	60 Grad Loop links oben
`CLoopN`	suffix, string, numeric	44 Grad Loop oben
`CLoopNE`	suffix, string, numeric	44 Grad Loop rechts oben
`CLoopE`	suffix, string, numeric	44 Grad Loop rechts
`CLoopSE`	suffix, string, numeric	44 Grad Loop rechts unten
`CLoopS`	suffix, string, numeric	44 Grad Loop unten
`CLoopSW`	suffix, string, numeric	44 Grad Loop links unten
`CLoopW`	suffix, string, numeric	44 Grad Loop links
`CLoopNW`	suffix, string, numeric	44 Grad Loop links oben
`LoopL`	suffix, numeric, string, numeric	linksdrehender 60 Grad Loop
`LoopR`	suffix, numeric, string, numeric	rechtsdrehender 60 Grad Loop
`CLoopL`	suffix, numeric, string, numeric	linksdrehender 44 Grad Loop
`CLoopR`	suffix, numeric, string, numeric	rechtsdrehender 44 Grad Loop

Tabelle 7. Zeichenbefehle für Transitionen 2

Name	Argumente	Erklärung
`EdgeL`	suffix, suffix, string, numeric	gerade Transition mit Label links
`EdgeR`	suffix, suffix, string, numeric	gerade Transition mit Label rechts
`ArcL`	suffix, suffix, string, numeric	15 Grad Bogen mit Label links
`ArcR`	suffix, suffix, string, numeric	15 Grad Bogen mit Label rechts
`LArcL`	suffix, suffix, string, numeric	30 Grad Bogen mit Label links
`LArcR`	suffix, suffix, string, numeric	30 Grad Bogen mit Label rechts
`VArcL`	suffix, suffix, numeric, numeric, string, numeric	Bogen mit Label links
`VArcR`	suffix, suffix, numeric, numeric, string, numeric	Bogen mit Label rechts
`VCurveL`	suffix, suffix, numeric, numeric, numeric, string, numeric	Kurve mit Label links
`VCurveR`	suffix, suffix, numeric, numeric, numeric, string, numeric	Kurve mit Label rechts
`ZZEdgeL`	suffix, suffix, string, numeric	Zickzack-Transition mit Label links
`ZZEdgeR`	suffix, suffix, string, numeric	Zickzack-Transition mit Label rechts

Tabelle 8. Zeichenbefehle für zusätzliche Label

Name	Argumente	Erklärung
`LabelL`	numeric, string	Label links an Transition schreiben
`LabelR`	numeric, string	Label rechts an Transition schreiben

Tabelle 9. Translation und Rotation

Name	Argumente	Erklärung
`TraRot`	pair, numeric	Zustände verschieben und rotieren
`TraRotEnd`		ohne Translation und Rotation
`SwivelLabel`		Label mit Zustand rotieren
`RigidLabel`		Label nicht mit Zustand rotieren

B Übersicht über die Parameter

Tabelle 10. Parameter für Zustände

Name	Erklärung
`StateLineColor`	Farbe der Zustandsumrandung
`StateLabelColor`	Farbe des Labels
`StateFillColor`	Farbe des Zustandsinneren
`StateLineStyle`	Art der Zustandsumrandung
`StateLineWidth`	Breite der Zustandsumrandung
`DimStateLineColor`	Farbe der Zustandsumrandung (gedimmt)
`DimStateLabelColor`	Farbe des Labels (gedimmt)
`DimStateFillColor`	Farbe des Zustandsinneren (gedimmt)
`DimStateLineStyle`	Art der Zustandsumrandung (gedimmt)
`DimStateLineCoef`	Koeffizient für Breite (gedimmt)
`StateLabelScale`	Koeffizient für Labelgröße (alle Zust.)
`StateFillStatus`	Art des Ausfüllens (alle Zust.)
`StateLineDblCoef`	Koeffizient für Linienbreite bei doppelter Umrandung
`StateLineDblSep`	Koeffizient für Abstand bei doppelter Umrandung

Tabelle 11. Parameter für Start- und Endpfeile

Name	Erklärung
`ArrowOnLargeState`	Pfeillänge bei großen Zuständen
`ArrowOnMediumState`	Pfeillänge bei mittleren Zuständen
`ArrowOnSmallState`	Pfeillänge bei kleinen Zuständen
`ArrowOnVerySmallState`	Pfeillänge bei sehr kleinen Zuständen

Tabelle 12. Parameter für Transitionen allgemein

Name	Erklärung
`EdgeLineColor`	Farbe der Transitionen
`EdgeLabelColor`	Farbe der Label
`EdgeLineStyle`	Zeichenstil der Transition
`EdgeLineWidth`	Linienbreite
`EdgeLabelScale`	Labelgröße
`DimEdgeLineColor`	Farbe der Transitionen (gedimmt)
`DimEdgeLabelColor`	Farbe der Label (gedimmt)
`DimEdgeLineStyle`	Zeichenstil der Transition (gedimmt)
`DimEdgeLineCoef`	Labelgröße (gedimmt)
`EdgeLineDblStatus`	Doppellinien (alle Transitionen)
`EdgeLineDblCoef`	Linienbreite bei Doppellinien
`EdgeLineDblSep`	Breite des Abstands bei Doppellinien
`EdgeLineBorderCoef`	Breite des Randes
`EdgeLineBorderColor`	Farbe des Randes
`EdgeArrowAngle`	Winkel der Pfeilspitze
`EdgeArrowLength`	Länge der Pfeilspitze

Tabelle 13. Parameter für Loops

Name	Erklärung
`LoopOnLargeState`	Loopgröße bei großen Zuständen
`LoopOnMediumState`	Loopgröße bei mittleren Zuständen
`LoopOnSmallState`	Loopgröße bei kleinen Zuständen
`LoopOnVerySmallState`	Loopgröße bei sehr kleinen Zuständen
`LoopAngle`	Öffnungswinkel für Loops
`CLoopAngle`	Öffnungswinkel für CLoops

Tabelle 14. Parameter für Bogen- und Zickzacktransitionen

Name	Erklärung
`ArcAngle`	Winkel des kleinen Bogens
`LArcAngle`	Winkel des großen Bogens
`ZZLineWidth`	Linienbreite der Zickzacktransitionen
`ZZSize`	Gesamtbreite der Zickzacktransitionen

Literatur

[Gas03] Paul Gastin. *GasTeX - Graphs and Automata Simplified in TeX*. 2003.

[Hob92] John D. Hobby. *A User's Manual for MetaPost*. AT&T Bell Laboratories, 1992.

[LS03] Sylvain Lombardy and Jacques Sakarovitch. *Vaucanson - A package for drawing automata and graphs*. 2003.

Index