Asymptote node.asy Example

Tao Wei
taowei@buffalo.edu

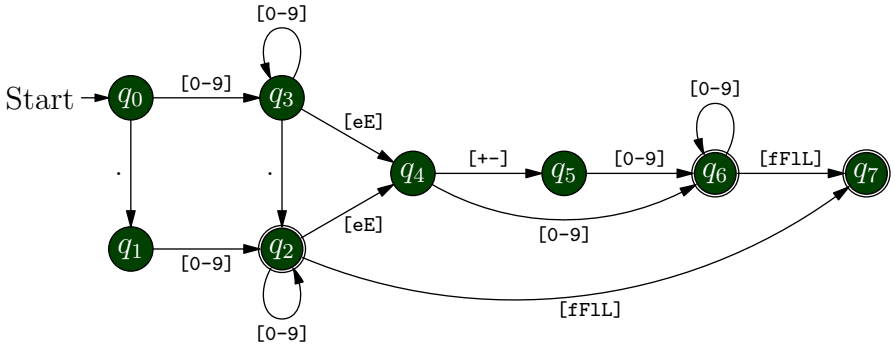
Contents

1 Introduction

node.asy is not only easy to use, but also with great extensibility. can draw graph theory, automata, flowchart, circuit-like graphics

2 Gallery

2.1 Automata

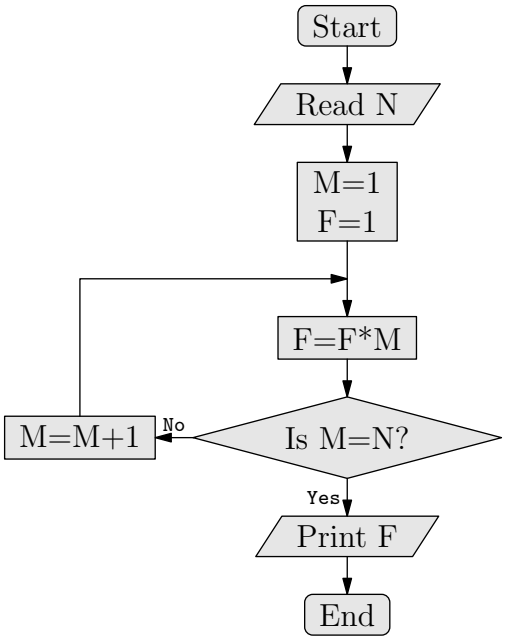



```
import node;

// define style
defaultnodestyle=nodestyle(textpen=white, drawfn=FillDrawer(
    darkgreen,black));
nodestyle ns2=nodestyle(textpen=white, drawfn=Filler(darkgreen
    )+DoubleDrawer(black));
nodestyle ns3=nodestyle(drawfn=None);
defaultdrawstyle=drawstyle(p=fontsize(8pt)+fontcommand("\
    ttfamily"), arrow=Arrow(6));

// define nodes
node q0=ncircle("$q_0$"),
     q1=ncircle("$q_1$"),
     q2=ncircle("$q_2$",ns2),
     q3=ncircle("$q_3$"),
     q4=ncircle("$q_4$"),
     q5=ncircle("$q_5$"),
     q6=ncircle("$q_6$",ns2),
     q7=ncircle("$q_7$",ns2),
     start=ncircle("Start",ns3);

// layout
defaultlayoutrel = false;
defaultlayoutskip = 2cm;
real u = defaultlayoutskip;

hlayout(0.6u, start, q0);
vlayout(q0, q1);
hlayout(q1, q2);
```

```
vlayout(-u, q2, q3);
layout(-30.0, q3, q4);
hlayout(q4, q5, q6, q7);

// draw nodes
draw(start,q0,q1,q2,q3,q4,q5,q6,q7);

// draw edges
draw(
        (q0--q1).l("."),
        (q1--q2).l("[0-9]"),
        (q3--q2).l("."),
        (q2--q4).l("[eE]"),
        (q0--q3).l("[0-9]").style("leftside"),
        (q3--q4).l("[eE]").style("leftside"),
        (q4--q5).l("[+-]").style("leftside"),
        (q5--q6).l("[0-9]").style("leftside"),
        (q6--q7).l("[fFlL]").style("leftside"),
        (q3..loop(N)).l("[0-9]"),
        (q2..loop(S)).l("[0-9]"),
        (q6..loop(N)).l("[0-9]"),
        (q4..bend..q6).l("[0-9]"),
        (q2..bend..q7).l("[fFlL]"),
        (start--q0)
    );
```

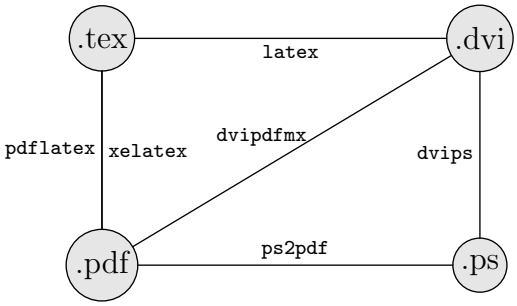
2.2 Flow Chart

```
import node;

// define style
defaultnodestyle=nodestyle(xmargin=4pt, ymargin=2pt, drawfn=
    FillDrawer(lightgray,black));
defaultdrawstyle=drawstyle(p=fontsize(8pt)+fontcommand("\
    ttfamily"), arrow=Arrow(6));

// define nodes
node start=nroundbox("Start"),
     read=nparallelogram("Read␣N"),
     b1=nbox(minipage2("M=1\\␣F=1")),
     b2=nbox("F=F*M"),
     d1=ndiamond("Is␣M=N?"),
     b3=nbox("M=M+1"),
     print=nparallelogram("Print␣F"),
```




```
15      end=nroundbox("End");
16
17  // layout
18  defaultlayoutskip=0.5cm;
19  vlayout(start, read, b1, new node, b2, d1, print, end);
20  hlayout(-defaultlayoutskip, d1, b3);
21
22  // draw nodes
23  draw(start, read, b1, b2, d1, b3, print, end);
24
25  // draw edges
26  draw(start--read, read--b1, b1--b2, b2--d1, (d1--print).l("Yes
        "),
27       (d1--b3).l("No"), print--end, b3--VH--middle(b1,b2));
```

2.3 Graph Illustration

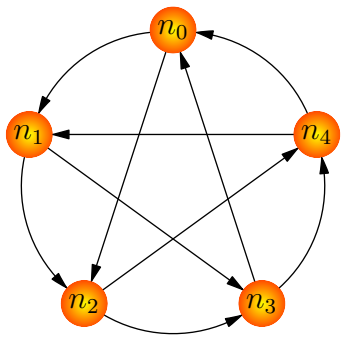



```
1  import node;
```

```
2
3  // define style
4  defaultnodestyle=nodestyle(drawfn=FillDrawer(lightgray,black))
       ;
5  defaultdrawstyle=drawstyle(p=fontsize(8pt)+fontcommand("\
       ttfamily"));
6
7  // define nodes
8  node[] n = ncircles(".tex", ".dvi", ".pdf", ".ps");
9
10 // layout
11 defaultlayoutrel=false;
12 gridlayout((2,2), (5cm, 3cm), n);
13
14 // draw nodes
15 draw(n);
16
17 // draw edges
18 draw(
19         (n[0]--n[1]).l("latex"),
20         (n[0]--n[2]).l("pdflatex"),
21         (n[0]--n[2]).l("xelatex").style("leftside"),
22         (n[1]--n[3]).l("dvips"),
23         (n[3]--n[2]).l("ps2pdf"),
24         (n[1]--n[2]).l("dvipdfmx")
25     );
```

2.4 Graph Theory

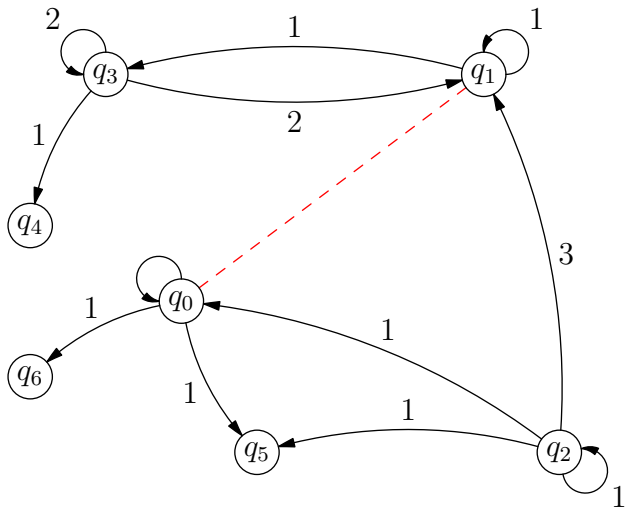
2.4.1 Simple Graph Theory




```
1  import node;
2
3  // define style
```

```
defaultnodestyle=nodestyle(drawfn=RadialShader(yellow,red));
defaultdrawstyle=drawstyle(p=fontsize(8pt)+fontcommand("\
    ttfamily"), arrow=Arrow(6));

// define nodes
node[] n = ncircles( "$n_0$",
        "$n_1$",
        "$n_2$",
        "$n_3$",
        "$n_4$");

// layout
circularlayout(2cm, startangle=90, n);

// draw nodes
draw(n);

// draw edges
draw(n[0]--n[2], n[2]--n[4], n[4]--n[1], n[1]--n[3], n[3]--n
    [0]);
draw(n[0]..bend..n[1], n[1]..bend..n[2], n[2]..bend..n[3],
     n[3]..bend..n[4], n[4]..bend..n[0]);
```

2.4.2 Graph Matrix Representation

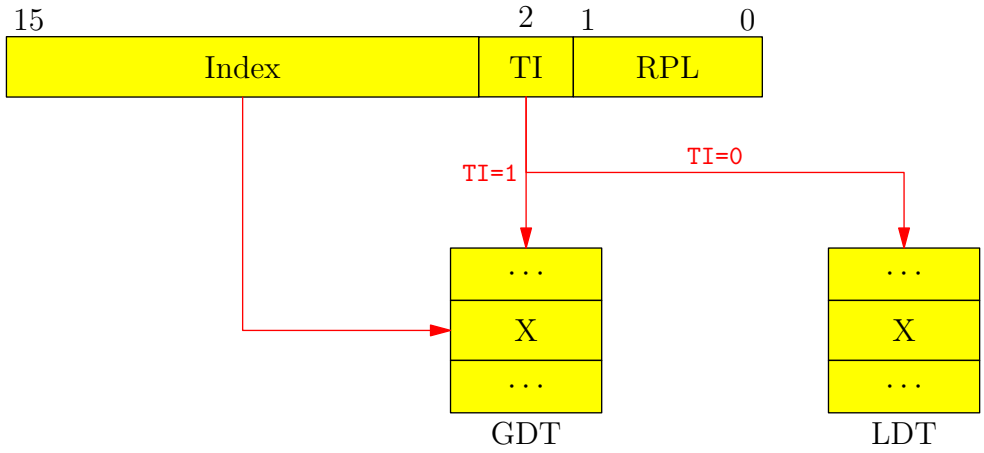



```
import nodegraph;

// define style
defaultdrawstyle=drawstyle(Arrow(6));
```

```
5
6  // define nodes
7  node[] ver=ncircles("$q_0$", "$q_1$", "$q_2$", "$q_3$", "$q_4$
       ", "$q_5$", "$q_6$");
8  pair[] pos={(0,0), (4,3), (5,-2), (-1,3), (-2,1), (1,-2),
       (-2,-1)};
9  real[][] matadj={{1,1,0,0,0,1,1},
10   {0,1,0,1,0,0},
11   {1,3,1,0,0,1},
12   {0,2,0,2,1,0}};
13
14 // layout
15 setpos(ver, pos*1cm);
16
17 // generate edges
18 edge[] edge=genedge(ver, matadj);
19 edge[edgeind(0,0,matadj)]=(ver[0]..loop(NW,90,1));
20 edge[edgeind(0,1,matadj)]=(ver[0]--ver[1]).style(drawstyle(p=
       red+dashed));
21
22 // draw nodes and edges
23 draw(edge);
24 draw(ver);
```

2.5 Boxes

2.5.1 Simple Boxes

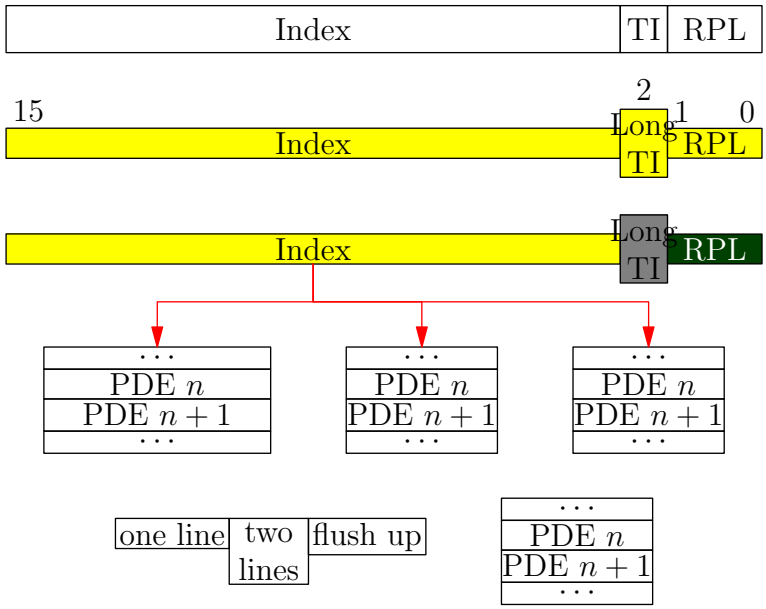


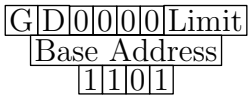

```
1  import node;
2
3  // define style
```

```
defaultnodestyle=nodestyle(ymargin=0.2cm, drawfn=FillDrawer(
    yellow));
defaultdrawstyle=drawstyle(p=red+fontsize(10pt)+fontcommand("\
    ttfamily"), arrow=Arrow);

// define nodes
node[] b = nboxes("Index", "TI", "RPL");
setwidth(10cm, new real[]{5,1,2} ... b);

node[] m = nboxes("$\cdots$", "X", "$\cdots$");
setwidth(2cm ... m);

node[] n = nboxes("$\cdots$", "X", "$\cdots$");
setwidth(2cm ... n);

// layout
defaultlayoutskip=0cm;
hlayout(... b);
vlayout(2cm, b[1], m[0]);
hlayout(3cm, m[0], n[0]);
vlayout(... m);
vlayout(... n);

// draw nodes
draw(concat(b, m, n));

// draw edges
draw(
        (b[0]--VH--m[1]),
        (b[1]--m[0]).l("TI=1"),
        (b[1]--VHV--n[0]).l("TI=0").style("leftside")
    );

// label
label("15", b[0]^NW, NE);
label("2", b[1]^N, N);
label("1", b[2]^NW, NE);
label("0", b[2]^NE, NW);
label("GDT", m[2]^S, S);
label("LDT", n[2]^S, S);
```

2.5.2 Fancy Boxes

```
import node;
```





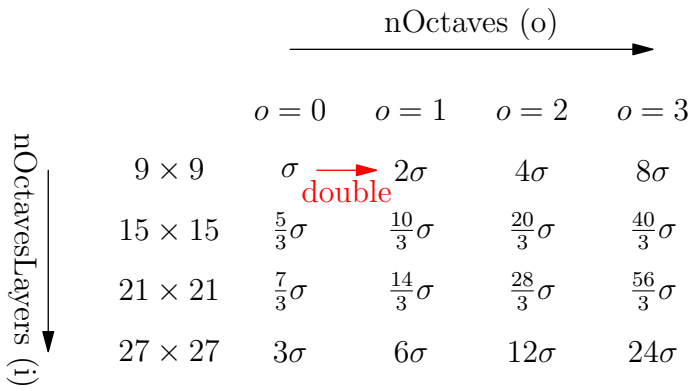

```
2
3  // define style
4  nodestyle ns1=nodestyle(drawfn=FillDrawer(yellow));
5  nodestyle ns2=nodestyle(drawfn=FillDrawer(gray));
6  nodestyle ns3=nodestyle(white, drawfn=FillDrawer(darkgreen));
7  nodestyle ns4=nodestyle(drawfn=FillDrawer(red));
8  nodestyle ns5=nodestyle(drawfn=FillDrawer(blue));
9  defaultdrawstyle=drawstyle(p=red+fontsize(10pt)+fontcommand("\
       ttfamily"), arrow=Arrow);
10
11
12 // define nodes
13 real h=gettextheight("Ag");
14 real u=10cm;
15
16 node[] a = nboxes("Index", "TI", "RPL");
17 setwidth(u, new real[]{13,1,2} ... a);
18 setheight(h ... a);
```

```
19
20 node b[] = nboxes(ns1, "Index", minipage2("Long␣\\␣TI"), "RPL"
       );
21 setwidth(u, new real[]{13,1,2} ... b);
22
23 node c[] = nboxes(new nodestyle[]{ns1, ns2, ns3}, "Index",
       minipage2("Long␣\\␣TI"), "RPL");
24 setwidth(u, new real[]{13,1,2} ... c);
25
26 hlayout(0 ...a);
27 vlayout(a[0], b[0]);
28 hlayout(0 ...b);
29 vlayout(b[0], c[0]);
30 hlayout(0 ...c);
31 draw(concat(a, b, c));
32
33 label("15", b[0]^NW, NE);
34 label("2", b[1]^N, N);
35 label("1", b[2]^NW, NE);
36 label("0", b[2]^NE, NW);
37
38 // define nodes
39 node[] pde = nboxes("$\cdots$", "PDE␣$n$", "PDE␣$n+1$", "$\
       cdots$");
40 node d = vpack(3cm ... pde);
41 node e = vpack(2cm ... pde);
42 node f = vpack(0 ... pde);
43 hcenterlayout(vlayout(2cm, middlen(...c), new node),
44               d, e, f);
45 draw(d, e, f);
46
47 // define nodes
48 node g = hpack(align=N
49           ... nboxes("one␣line", minipage2("two␣\\␣lines"), "
                flush␣up"));
50 node h = vpack(align=W ... pde);
51 hcenterlayout(vlayout(2cm, middlen(d,e,f), new node),
52           g, h);
53 draw(g,h);
54
55 // define nodes
56 node i1 = hpack(... nboxes("G","D","0","0","0","0","Limit"));
57 node i2 = nbox("Base␣Address");
58 node i3 = hpack(... nboxes("1","1","0","1"));
59 node i = vpack(i1, i2, i3);
```

```
node j1 = hpack(... nboxes(new nodestyle[]{ns1,ns1,ns2,ns2,ns2
    ,ns2,ns3},
            "G","D","0","0","0","0","Limit"));
node j2 = nbox("Base Address", ns4);
node j3 = hpack(... nboxes(new nodestyle[]{ns5}, "1","1","0","
    1"));
node j = vpack(j1,j2,j3);
vlayout(2cm, middlen(g,h), i);
vlayout(i,j);
draw(i,j);


// draw edges
draw(
        (c[0]--VHVd(0.5cm)--d),
        (c[0]--VHVd(0.5cm)--e),
        (c[0]--VHVd(0.5cm)--f)
    );
```

2.6 Scientific Publication

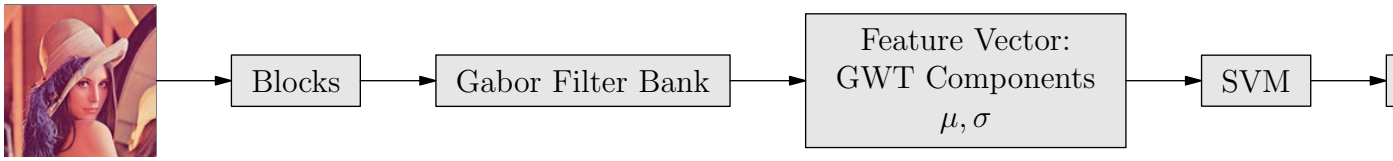
2.6.1 Text Annotation




```
import node;

// define style
defaultdrawstyle=drawstyle(arrow=Arrow);
drawstyle es2 = drawstyle(p=red, arrow=Arrow);

// define nodes
Label[][] strs = {
    {"", "$o=0$", "$o=1$", "$o=2$", "$o=3$"},
```

```
    {"$9\times 9$", "$\sigma$", "$2\sigma$", "$4\sigma$", "$8\
        sigma$"},
    {"$15\times 15$", "$\frac{5}{3}\sigma$", "$\frac{10}{3}\
        sigma$", "$\frac{20}{3}\sigma$", "$\frac{40}{3}\sigma$"
        },
    {"$21\times 21$", "$\frac{7}{3}\sigma$", "$\frac{14}{3}\
        sigma$", "$\frac{28}{3}\sigma$", "$\frac{56}{3}\sigma$"
        },
    {"$27\times 27$", "$3\sigma$", "$6\sigma$", "$12\sigma$",
        "$24\sigma$"}
};
node[][] n = node(strs);

node r1, r2, c1, c2; // dump nodes

// layout
defaultlayoutrel=false;
pair xyskip = (2cm, 1cm) * 0.8;
gridlayout(xyskip, n);
vlayout(-xyskip.y, n[0][1], r1);
vlayout(-xyskip.y, n[0][4], r2);
hlayout(-xyskip.x, n[1][0], c1);
hlayout(-xyskip.x, n[4][0], c2);

// draw nodes
draw(n);

// draw edges
draw(
        (r1--r2).l("nOctaves (o)").style("autorot").style("
            leftside"),
        (c1--c2).l("nOctavesLayers (i)").style("autorot"),
        (n[1][1]--n[1][2]).l("double").style(es2).shorten(5,5)
    );
```

2.6.2 Image Node



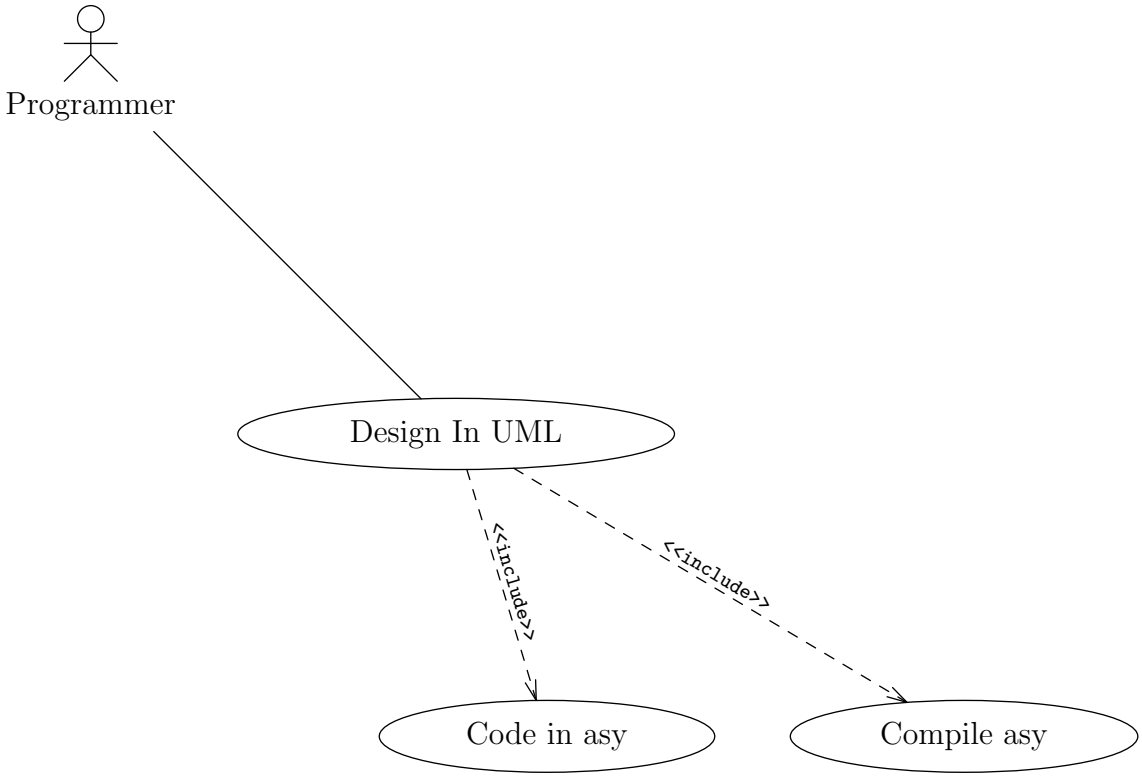

```
import nodeimage;
```

```
import math;

// define style
defaultnodestyle=nodestyle(xmargin=6pt, ymargin=4pt, drawfn=
    FillDrawer(lightgray,black));
defaultdrawstyle=drawstyle(p=fontsize(8pt)+fontcommand("\
    ttfamily"), arrow=Arrow(6));

node img = nimage("lena.jpg", (2cm,2cm));
node[] flow = nboxes("Blocks",
        "Gabor␣Filter␣Bank",
        minipage2("Feature␣Vector:␣\\␣GWT␣Components␣\\␣$\mu,\
            sigma$"),
        "SVM",
        "Class");

hlayout(img ... flow);

draw(img ... flow);
draw(
        (img--flow[0]),
        (flow[0]--flow[1]),
        (flow[1]--flow[2]),
        (flow[2]--flow[3]),
        (flow[3]--flow[4])
    );
```

2.7 SML

2.7.1 SML Hello

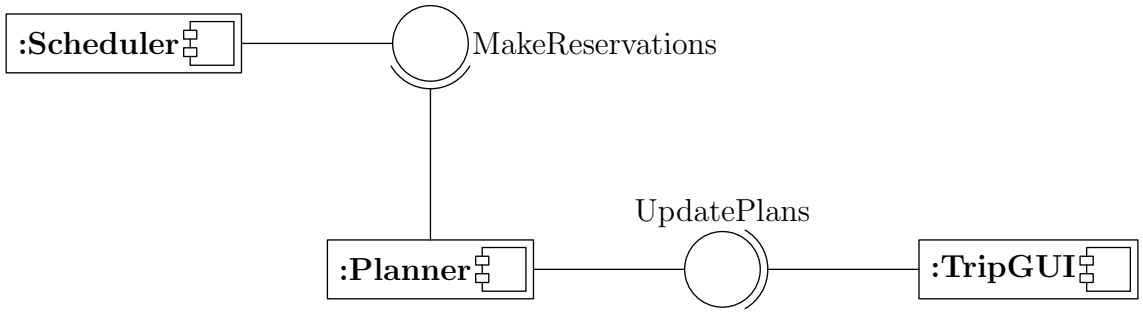
```
import nodesml;

// define style
defaultnodestyle=nodestyle(mag=1.4);
defaultdrawstyle=drawstyle(align=LeftSide, p=fontsize(8pt)+
    fontcommand("\ttfamily")+dashed, Arrow(SimpleHead));
drawstyle es2=drawstyle(p=fontsize(8pt)+fontcommand("\ttfamily
    "));

// define nodes
node a=sml_actor("Programmer");
node c=nellipse("Design␣In␣UML");
node c1=nellipse("Code␣in␣asy");
node c2=nellipse("Compile␣asy");
```




```
13
14 // layout
15 real u=1cm, v=5cm;
16 layout(-45.0, v, a, c, nodeph);
17 hcenterlayout(nodeph, u, c1, c2);
18
19 // draw nodes
20 draw(a, c, c1, c2);
21
22 // draw edges
23 draw(
24     (a--c).style(es2),
25     (c--c1).l("<<include>>").style("autorot"),
26     (c--c2).l("<<include>>").style("autorot")
27 );
```

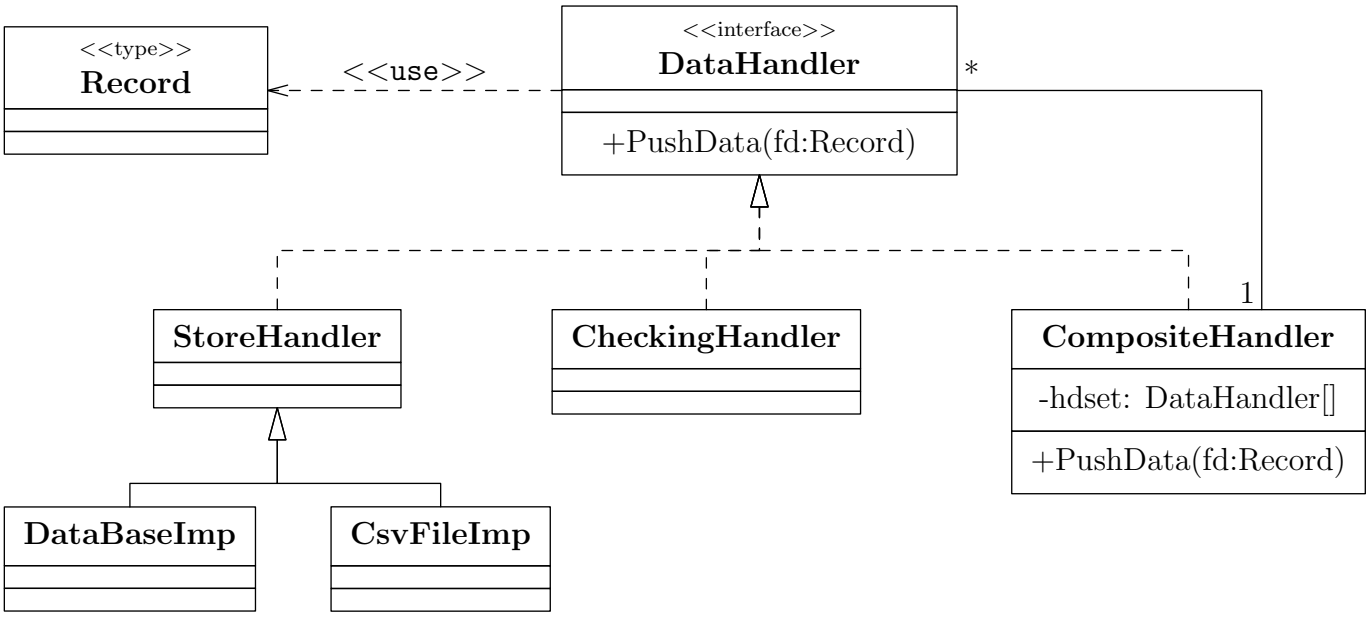
2.7.2 SML Component

```
1 import nodesml;
2
3 node a=sml_com(":Scheduler");
4 node b=sml_iball("MakeReservations", S);
5 node c=sml_com(":Planner");
6 node d=sml_iball("UpdatePlans", E);
```




```
node e=sml_com(":TripGUI");

defaultlayoutskip=2cm;
hlayout(a,b);
vlayout(b,c);
hlayout(c,d,e);

draw(a, b, c, d, e);

draw(a--b, b--c, c--d, d--e);
```

2.7.3 SML Class

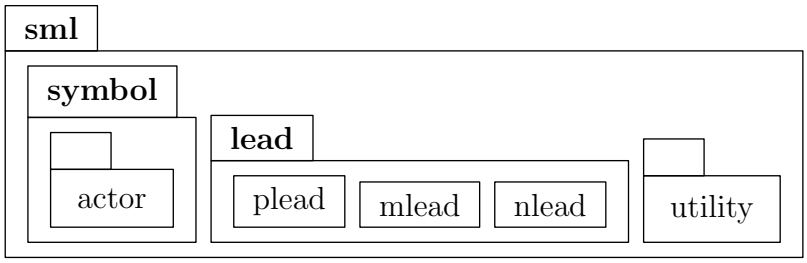



```
import nodesml;

// define nodes
node record=sml_class("Record", "type");
node datah=sml_class("DataHandler", "interface", "", "+
    PushData(fd:Record)");
```

```
node storeh=sml_class("StoreHandler");
node checkh=sml_class("CheckingHandler");
node comph=sml_class("CompositeHandler","","-hdset:␣
    DataHandler[]", "+PushData(fd:Record)");
node dbi=sml_class("DataBaseImp");
node cfi=sml_class("CsvFileImp");

// layout
hlayout(1cm, dbi, cfi);
vlayout(-2cm, middlen(dbi, cfi), storeh);
hlayout(2cm, storeh, checkh, comph);
flush(N, storeh, checkh, comph);
vlayout(-3cm, middlen(storeh, checkh, comph), datah);
hlayout(-1cm, datah, record);
flush(W, dbi, record);

draw(dbi, cfi, storeh, checkh, comph, datah, record);

// draw edges
drawstyle es2=drawstyle(p=dashed+fontcommand("\ttfamily"),
    Arrow(SimpleHead));
drawstyle es3=drawstyle(p=dashed, BeginArrow(12,NoFill));
drawstyle es4=drawstyle(BeginArrow(12,NoFill));

draw(
    (datah--record).l("$<<$use$>>$").style(es2),
    (datah--VHVd(1cm)--storeh).style(es3),
    (datah--VHVd(1cm)--checkh).style(es3),
    (datah--VHVd(1cm)--comph).style(es3),
    (storeh--VHVd(1cm)--dbi).style(es4),
    (storeh--VHVd(1cm)--cfi).style(es4),
    (datah--HV--node(pos=comph^NNE))
);


// label
label("*",datah^E,NE);
label("1",comph^NNE,NW);
```

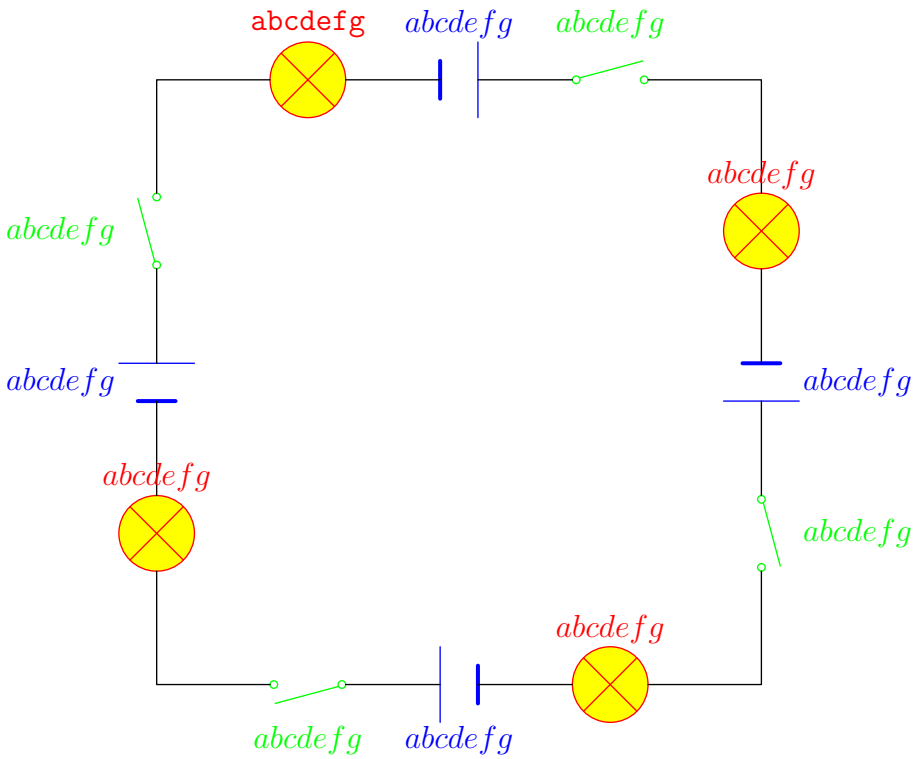
2.7.4 SML Lead

```
import nodesml;

node c1=sml_lead("symbol", sml_lead("", "actor"));
```




```
node c2=sml_lead("lead", "plead", "mlead", "nlead");
node c3=sml_lead("", "utility");
node cc=sml_lead("sml", c1, c2, c3);

draw(cc);
```

2.8 Circuit




```
import nodecircuit;

// define style
defaultcircuitlightstyle=nodestyle(textpen=red+fontcommand("\
    ttfamily"), filler=Filler(yellow), indrawer=red, outdrawer=
    red);
defaultcircuitbatterystyle=nodestyle(textpen=blue+fontcommand(
    "\ttfamily"), indrawer=InDrawer(blue,1,3));
```

```
defaultcircuitswitchstyle=nodestyle(textpen=green+fontcommand(
    "\ttfamily"), indrawer=green);

// define nodes
node l1=circuit_light("abcdefg", N);
node l2=circuit_light("$abcdefg$", E);
node l3=circuit_light("$abcdefg$", S);
node l4=circuit_light("$abcdefg$", W);
node b1=circuit_battery("$abcdefg$", E);
node b2=circuit_battery("$abcdefg$", S);
node b3=circuit_battery("$abcdefg$", W);
node b4=circuit_battery("$abcdefg$", N);
node s1=circuit_switch("$abcdefg$", N);
node s2=circuit_switch("$abcdefg$", E);
node s3=circuit_switch("$abcdefg$", S);
node s4=circuit_switch("$abcdefg$", W);

node n1, n2, n3, n4; // dump nodes

// layout
defaultlayoutrel=false;
real u=2cm, v=2cm;
hlayout(u, n1, l1, b1, s1, n2);
vlayout(v, n2, l2, b2, s2, n3);
hlayout(-u, n3, l3, b3, s3, n4);
vlayout(-v, n4, l4, b4, s4);

// draw nodes
draw(l1,l2,l3,l4,b1,b2,b3,b4,s1,s2,s3,s4);

// draw edges
draw(l1--b1, b1--s1, s1--HV--l2, l2--b2, b2--s2, s2--VH--l3,
    l3--b3, b3--s3,
     s3--HV--l4, l4--b4, b4--s4, s4--VH--l1);
```

3 Usage

A typical use of this library involves 4 steps:

1. define node and edge styles
2. define nodes
3. layout and draw nodes
4. draw edges

3.1 define node and edge styles

this step can be ignored if the default black-white style satisfies you, or a single line if you want edges to have arrows:

```
defaultdrawstyle=drawstyle(Arrow);
```

there are two global variables to control the default node style and edge style, you can overwrite them to satisfy your own needs:

```
defaultnodestyle=nodestyle(drawfn=FillDrawer(lightgray,black));
defaultdrawstyle=drawstyle(p=fontsize(8pt)+fontcommand("\ttfamily"), arrow=Arrow(6));
```

you can define your own styles if you want multiple styles:

```
defaultnodestyle=nodestyle(textpen=white, drawfn=FillDrawer(darkgreen,black));
nodestyle ns2=nodestyle(textpen=white, drawfn=Filler(darkgreen)+DoubleDrawer(black));
nodestyle ns3=nodestyle(drawfn=None);
```

3.2 define nodes

shaped nodes, (nbox can be ncircle, nellipse, nroundbox, ndiamond, nparallelogram)

```
node n0 = nbox("$n_0$");
node n1 = ncircle("$n_1$", ns2);
```

or multiple nodes of the same shape and style (nboxes, ..., nparallelograms)

```
node[] n = nboxes("$n_0$", "$n_1$", "$n_2$", "$n_3$", "$n_4$");
node[] n = ncircles(ns2, "$n_0$", "$n_1$", "$n_2$", "$n_3$", "$n_4$");
```

text nodes (node)

```
node n0 = node("$n_0$");
node[] n = node("$n_0$", "$n_1$", "$n_2$", "$n_3$", "$n_4$");
Label[][] strs = {
{"$a_{0,0}$", "$\cdots$", "$a_{0,n}$"},
{"$\vdots$", "$\ddots$", "$\vdots$"},
{"$a_{m,0}$", "$\cdots$", "$a_{m,n}$"}};
node[][] n2d = node(strs);
```

node resize (setsize, setwidth, setheight):

```
setwidth(... nds); // automatically set to the largest width
setwidth(w=width ... nds); // set width to "w", if w=0, automatically
```

```
setheight(h ... nds);
setsize((w,h) ... nds);
// set nds total width to 10cm, with ratios of 5:1:2
setwidth(10cm, new real[]{5,1,2} ... nds);
```

compound nodes (npack, hpack, vpack), nodes can be firsted resized, and then layouted before packing:

```
hpack(h=-1 ... nds); // do not reisze nodes
hpack(h=0 ... nds); // nodes are set to same height
hpack(h=height, ... nds); // nodes are set to "height" (
   positive)
vpack(w=<-1, 0 or width> ... nds)
```

symbol nodes, please refer to nodecircuit, and nodesml.

3.3 layout nodes

two global variables to control the layout properties

- relative layout: layout according to the outline of nodes
- absolute layout: layout accordng to the position of nodes

```
defaultlayoutrel = true; // relative layout
defaultlayoutskip = 1cm; // nodes skip
```

layouts (there is a global node "nodeph" as a place holder):

```
layout(dir ... nds);
hlayout(... nds);
vlayout(... nds);
gridlayout((rownum, colum) ... nds);
gridlayout(nds2d);
circularlayout(... nds);
centerlayout(dir, refnode ... nds);
hcenterlayout(refnode ... nds);
vcenterlayout(refnode ... nds);
```

set to absolute positions:

```
setpos(nds, positions);
```

flush:

```
flush(align ... nds); // align can be N, S, E, W
```

layout APIs reference:

```
layout(dir, skip, rel ... nds);
hlayout(skip, rel ... nds);
vlayout(skip, rel ... nds);
```

```
gridlayout((rownum, colnum), roworder, (xskip, yskip), rel ...
    nds);
circularlayout(radius, center, startangle, angleskip ... nds);
centerlayout(dir, refnode, skip, rel ... nds);
hcenterlayout(refnode, skip, rel ... nds);
vcenterlayout(refnode, skip, rel ... nds);
```

3.4 draw nodes

```
draw(... nds);
```

3.5 draw edges

```
draw(
        (n1--n2),
        (n1..n2),
        (n1..loop(N)),
        (n1..bend..n2),
        (n1--n2).l("label"),
        (n1--n2).l("label").style(es2),
        (n1--n2).l("label").style("autorot"),
        (n1--n2).l("label").style("leftside"),
        (n1--n2).l("label").shorten(5,5)
);
```