The Expressg METAPOST package for drawing box-line-annotation diagrams^{*}

Peter Wilson Catholic University of America Now at peter.r.wilson@boeing.com

> (First published 1996/05/09) 2004/03/17

Abstract

The expressg METAPOST package provides facilities to assist in drawing diagrams that consist of boxes, lines, and annotations. Particular support is provided for creating EXPRESS-G diagrams

Contents

1	Intr	oduction	2
2	The	expressg package	3
	2.1	Internal variables	3
	2.2	Routines	5
		2.2.1 Utility routines	6
		2.2.2 Path routines	7
		2.2.3 Line drawing routines	8
		2.2.4 Box drawing routines	10
	2.3	Extra BLA variables and routines	13
3 An example		example	17
	3.1	Diagram 1: Some boxes and lines	18
	3.2	Diagrams 2–5: EXPRESS-G like models	24
		3.2.1 Diagram 2: Schema level diagram	25
			28
		3.2.3 Diagram 4: Supertypes and subtypes	32
		3.2.4 Diagram 5: Portion of larger model	36
	3.3		41
		-	42

*This file (expressg.dtx) has version number v1.61, last revised 2004/03/17.

		3.3.2 Diagram 7: Shlaer-Mellor
		3.3.3 Diagram 8: IDEF1X
		3.3.4 Diagram 9: OMT
		3.3.5 Diagram 10: E-R
		3.3.6 Diagram 11: NIAM 52
	3.4	Comments
	3.5	Running the METAPOST example file
		3.5.1 Using pdfIAT _E X
	3.6	Using $\square T_E X$ instead of $T_E X$
	3.7	Using PostScript fonts
4	The	MetaPost code 61
	4.1	Variables
	4.2	Utility routines
	4.3	Path routines
	4.4	Line end drawing routines
	4.5	Line drawing routines
	4.6	Box drawing routines
	4.7	Extra BLA variables and routines

List of Figures

1	Length parameters for arrowhead and fanin line end styles	4
2	Parameters for (inset) boxes	5
3	Results of the VH, VhV and HvH routines $\ldots \ldots \ldots \ldots \ldots \ldots$	6
4	Results of the $\tt VyV$ and $\tt HxH$ routines $\hfill\hfilt\hfill\h$	7
5	Calculated points on rectangular boxes	10
6	Calculated points on rhomboidal boxes	13
7	Length parameters for diamond line end styles	14
8	Calculated points on diamond boxes	15
9	Calculated points on a drum	17

1 Introduction

EXPRESS-G is an ISO international standard graphical information modeling language [ISO94, SW94], and is a subset of the EXPRESS lexical object-flavoured information modeling language. Like most graphical modeling languages, EXPRESS-G diagrams consist of boxes, lines, and annotations associated with the boxes and the lines. As a colleague of mine once put it, these kinds of graphical languages are BLA (boxes, lines, annotations) with respect to lexical languages.

METAPOST [Hob92] is a powerful lexical language for producing Encapsulated PostScript¹ diagrams, and is based on the METAFONT [Knu86] language for creating fonts.

 $^{^1\}mathrm{PostScript}$ is a registered trademark of Adobe Systems Incorporated.

The expressg package provides extensions to the base METAPOST language to facilitate the creation of EXPRESS-G diagrams. The package may also be used as is, or certainly by extension, for the creation of other BLA diagrams.

METAPOST is typically used to generate pictures for inclusion in LATEX documents [Lam94]. By default it uses TEX for typesetting any text; this behaviour can be modified so that LATEX is used instead. If standard PostScript fonts (like Times) are used, then LATEX typesetting may be avoided altogether. Running METAPOST and methods of including METAPOST diagrams in (pdf)LaTeX documents are described in [Wil99].

The expressg package has been some years in the making. For occasional diagrams I got by with using either the simple IATEX picture commands, or general GUI drawing packages. As time went on and the number and complexity of the diagrams increased it became, to me at least, more important to use a robust and powerful drawing language than fancy GUI packages. This was especially important if a diagram had to be changed at a later date, as the language denoted exactly how the drawing had been produced. With the GUI tools all that you had was the picture and whatever graphics file format the tool used — there was no user-sensible record of how any result was achieved. Also, generally speaking I find it quicker to use METAPOST than a GUI tool as too much time has to be spent using a mouse or a cursor to reposition and rescale the drawing elements.

Section 2 describes the facilities provided by the package. Several worked example diagrams are presented in Section 3, together with information on how to run METAPOST. Commented code for the package macros² is in Section 4.

It is assumed that the reader has some understanding of METAPOST, although the example in Section 3 does provide an *aide-memoire* for most of the METAPOST constructs that are likely to be used in creating BLA diagrams. The definitive METAPOST manual is [Hob92] but both [Hoe98] and [GRM97] give some interesting examples of how to use it; in fact the front cover of Hoenig's book is a METAPOST picture. Similarly, some familiarity with BLA diagramming would be of assistance.

I suggest that on first reading it may be sensible to skim Section 2 first and then peruse the example in Section 3 more closely. After this go back and re-read Section 2 as it then should make more sense.

This manual is typeset according to the conventions of the IAT_EX DOC-STRIP utility which enables the automatic extraction of the IAT_EX macro source files [GMS94].

2 The expressg package

2.1 Internal variables

The package includes several internal variables that are initialised to commonly appropriate values. This section describes the principal internal variables. Short

²If requested, I may be prepared to consider providing additional macros, but no promises.



Figure 1: Length parameters for arrowhead and fanin line end styles

descriptions of the others, which are likely to be used in diagrams only very occasionally, can be found in Section 4.

The values of internal variables in METAPOST can only be changed via assignment. For example:

internal_variable := expression;

u	u is the value of the unit of length for a diagram; it is initialised to 1mm.				
maxx					
maxy	maxy are the maximum x and y coordinate values for a diagram. These are ini-				
tialised to the maximum size for a diagram within an ISO standard of					
(i.e., maxx:=159.5u and maxy:=210u).					
The next set of variables control the thickness of lines and the dime					
	line end styles.				
normalpen	Lines with normal linethickness are drawn with normalpen.				
thickpen	Thick lines are drawn with thickpen.				
thinpenpen	Thin lines are drawn with thinpen.				
dotspen	Dotted lines are drawn with dotspen.				
dotpen					
dotdiam	· · · · · · · · · · · ·				
smoothrad					
straight lines.					
drumlid	drumlid is the ratio of the minor to major diameter of the ellipse forming the				
	top of the drum box; the larger the value the more the apparent viewpoint shifts				
	to above the drum. The default value is 0.2.				
gal	An arrowhead is defined by the height of the triangle and the base of the				
gab	triangle forming the arrowhead, as shown in Figure 1. gal and gab are the length				
	and base width of arrows for arrowed line end styles.				
gfl Like an arrowhead, a fanin is defined by the height and the base					
gfb triangular fan shape, as shown in Figure 1. gfl and gfb are the length					
	width of fans for fanin line end styles. They are initialised to the default values.				
	The next set of variables control certain geometric aspects of the various				
	EXPRESS-G boxes.				



Figure 2: Parameters for (inset) boxes

onelineh is the minimum height of a box that encloses a single line of text. onelineh This is initialised to 5u. The length (sdtbl), height (sdtbh) and line inset (sdtbs) for simple data sdtbl type boxes (e.g., a BOOLEAN data type box), as illustrated in Figure 2. These sdtbh are initialised as 22u, onelineh and 2u respectively. sdtbs The length (sdtebl), height (sdtebh) and line inset (sdtebs) for a simple sdtebl data type EXPRESSION box. The value of sdtebl is 28u, and the other values sdtebh sdtebs are the same as for the other simple data type boxes. The length (sdtbgel), height (sdtbgeh) and inset for GENERICENT data sdtbgel type boxes. The value of sdtbgel is 38u, and the other values are the same as for sdtbgeh the other simple data type boxes. sdtbges The length (pcon1) of an average numeric (proxy) page connector (e.g., for pconl (9,9 (9,9))) and the height (pconh) of an average page connector, which is pconh initialised for one line of text. enth The heights of an average entity (enth), type — including enumeration and select — (typeh), and schema (schemah) boxes. These are all initialised to cater typeh schemah for one line of text. ish is the height of an interschema reference box with one line of text. isrh ish isrh is the height of an interschema reference box with a rename, where there is one line of text each for the original and renamed names. eventh The height (eventh) of a one-lined event box, and the slope (eventslope) of the sides of an event box (e.g., 0.25 or 1/4). eventslope The next set of variables control the amount of space surrounding text on EXPRESS-G diagrams. nextra is the margin around names when put into entity, schema, interschema nextra reference, page connector, and event boxes. niextra is the margin around names when put into type boxes. niextra ndextra is the margin around names when used on attribute lines. ndextra 2.2Routines The principal aspects of an EXPRESS-G diagram are lines and line ends of vari-

ous styles, and boxes of several different kinds. Many routines are provided for

5



Figure 3: Results of the VH (left), VhV (middle) and HvH (right) routines

positioning and drawing these on a diagram.

The following sections describe most of the routines. Short descriptions of the others, which are effectively only used as internal support routines, can be found in Section 4.

A note on naming conventions used in describing the routines:

- As usual with METAPOST, a name starting with z is a point, and names staring with x or y are x- and y-coordinates respectively.
- The letters i through n denote (point, coordinate) suffixes. Suffixes are usually a number, such as 37, but may also have alphanumeric characters after the number (e.g., 37C2D).

2.2.1 Utility routines

Several utility routines are provided defining some shortcuts and to ease some of the calculations that may be required when producing a diagram.

- VH VH(j, k) calculates the intersection point, zjvh, between the vertical line through the point zj and the horizontal line through the point zk, as shown in Figure 3.
- VhV VhV(j, k) calculates the point zjvhv on the vertical line through the point zj and the point zkvhv on the vertical line through the point zk, such that the line zjvhv--zkvhv is horizontal and centered vertically between the two given points, as shown in Figure 3.
- HvH HvH(j, k) calculates the point zjhvh on the horizontal line through the point zj and the point zkhvh on the horizontal line through the point zk, such that the line zjhvh--zkhvh is vertical and centered horizontally between the two given points, as shown in Figure 3.
- VyV VyV(j, i, k) is a generalisation of the VhV routine. It calculates the point zjvyv which is on the vertical line through zj and the horizontal line through yi,





and the point zkvyv which is on the vertical line through zk and the horizontal line through yi. An example result is shown in Figure 4.

HxH(j, i, k) is a generalisation of the HvH routine. It calculates the point zjhxh which is on the horizontal line through zj and the vertical line through xi, and the point zkhxh which is on the horizontal line through zk and the vertical line through xi. An example result is shown in Figure 4.

namespace($\langle name \rangle$) ($\langle margin \rangle$) calculates the length of the text string $\langle name \rangle$ plus the $\langle margin \rangle$ length. You may use it like:

; =	3/2names	pace(btex	name	etex)	(4u))
-----	----------	-----------	------	-------	------	---

dashes is shorthand for a dashed linestyle. Use it like: draw ... dashes;.
dots is shorthand for a dotted linestyle. Use it like: draw ... dots;.

drawgrid draws a 16 by 21cm, 1cm spaced grid composed of thin dashed lines. Labels are typeset giving the distance of the lines from the bottom left corner of the grid in millimetres.

2.2.2 Path routines

The package provides some general path making routines.

sharply(zi, zj, ... zn) will generate a piecewise linear path consisting of the straight lines from point zi to point zj ... to point zn. It may be used like: draw sharply(z1, z2, z3, z4);

to draw a line through the given points.

smoothly

Like the sharply routine, smoothly(zi, zj, ... zn) will generate a piecewise linear path consisting of the straight lines from point zi to point zj ... to point zn, except that the corners of the path at the interior points are rounded with an arc of radius smoothrad. It may be used like:

draw smoothly(z1, z2, z3, z4) dashes;

to draw a dashed line with rounded corners at the given points. Or if a dotted line is needed:

pickup dotspen;

namespace

HxH

dashes dots drawgrid

sharply

```
draw smoothly(z1, z2, z3, z4) dots;
pickup normalpen;
```

Like the & operator, the binary ~ operator concatenates two paths replacing the sharp join with a circular arc of radius smoothrad.

2.2.3 Line drawing routines

Routines are provided for drawing a variety of lines with differing line ending styles. Line styles include dotted, dashed, and continuous, and line thicknesses may be normal or thick. Line end styles include open circles, black dots, open and closed (black) arrowheads, and a fanin style.

The final uppercase characters of a line drawing routine indicate the kind of line end style that will be used. The character O (uppercase O, not zero) is for an Open circle, D is for a black Dot, OA is for an Open Arrowhead, CA is for a Closed (black) Arrowhead, and F is for a Fanin.

draw0	The line ending drawing routines are described first. draw0(i, j) draws an				
	open circle, diameter dotdiam, at the zj end of the vector from zi to zj.				
drawD	drawD(i, j) draws a black dot, diameter dotdiam, at the zj end of the vector				
	from zi to zj.				
drawOA	drawOA(i, j) draws an open arrowhead, length gal and base width gab, at				
	the zj end of the vector from zi to zj.				
drawCA	drawCA(i, j) draws a closed (black) arrowhead, length gal and base width				
	gab, at the zj end of the vector from zi to zj.				
drawF	drawF(i, j) draws a fanin, length gfl and base width gfb, at the zj end of				
	the vector from zi to zj.				
	The line drawing routines are essentially prepackaged versions of different com-				
	binations of the path and line ending routines.				
drawdots	drawdots(i, j) draws the dotted line zizj.				
drawdotsthree	drawdotsthree(i, j, k) draws the dotted piecewise linear line zizjzk.				
drawdotsfour	drawdotsfour(i, j, k, 1) draws the dotted piecewise linear line zizjzkzl.				
drawdots0	drawdotsO(i, j) draws the dotted line zizj, ending in an open circle,				
	diameter dotdiam, at zj.				
drawdotsthree0	drawdotsthreeO(i, j, k) draws the dotted piecewise linear line zizjzk.				
	ending in an open circle, diameter dotdiam, at zk.				
drawdotsfourO	drawdotsfourO(i, j, k, 1) draws the dotted piecewise linear line zizjzkzl,				
	ending in an open circle, diameter dotdiam, at zl.				
drawdots00	drawdotsOO(i, j) draws the dotted line zizj, ending in open circles, di-				
	ameter dotdiam, at zi and zj.				
drawdash	drawdash(i, j) draws the dashed line zizj.				
drawdashthree	drawdashthree(i, j, k) draws the dashed piecewise linear line zizjzk.				
drawdashfour	our drawdashfour(i, j, k, l) draws the dashed piecewise linear line zizjzkzl				
drawdash0	drawdashO(i, j) draws the dashed line zizj, ending in an open circle,				
	diameter dotdiam, at zj.				
drawdashthree0	drawdashthreeO(i, j, k) draws the dashed piecewise linear line zizjzk.				

	ending in an open circle, diameter dotdiam, at zk.
drawdashfourO	drawdashfourO(i, j, k, 1) draws the dashed piecewise linear line zizjzkzl.
	ending in an open circle, diameter dotdiam, at zl.
drawdash00	drawdash00(i, j) draws the dashed line zizjzkzl. ending in open
	circles, diameter dotdiam, at zi and zj.
drawnormal	drawnormal(i, j) draws the normal thickness line zizj.
drawnormalthree	drawnormalthree(i, j, k) draws the normal thickness piecewise linear line
	zizjzk.
drawnormalfour	drawnormalfour(i, j, k, l) draws the normal thickness piecewise linear
	line zizjzkzl.
drawnormal0	drawnormalO(i, j) draws the normal thickness line zizj, ending in an
	open circle, diameter dotdiam, at zj.
drawnormalthreeO	drawnormalthreeO(i, j, k) draws the normal thickness piecewise linear line
	zizjzk, ending in an open circle, diameter dotdiam, at zk.
drawnormalfourO	drawnormalfourO(i, j, k, 1) draws the normal thickness piecewise linear
	line zizjzkzl, ending in an open circle, diameter dotdiam, at zl.
drawnormal00	drawnormal00(i, j) draws the normal thickness line zizj, ending in open
	circles, diameter dotdiam, at zi and zj.
drawnormalD	drawnormalD(i, j) draws the normal thickness line zizj, ending in a black
	dot, diameter dotdiam, at zj.
drawnormalthreeD	drawnormalthreeD(i, j, k) draws the normal thickness piecewise linear line
	zizjzk, ending in a black dot, diameter dotdiam, at zk.
drawnormalfourD	drawnormalfourD(i, j, k, 1) draws the normal thickness piecewise linear
	line zizjzkzl, ending in a black dot, diameter dotdiam, at zl.
drawnormalDD	drawnormalDD(i, j) draws the normal thickness line zizj, ending in black
	dots, diameter dotdiam, at zi and zj.
drawnormalCA	drawnormalCA(i, j) draws the normal thickness line zizj, ending in a
	closed (black) arrowhead, length gal and base width gab, at zj.
drawnormalthreeCA	drawnormalthreeCA(i, j, k) draws the normal thickness piecewise linear
	line $zi-zj-zk$ ending in a closed (black) arrowhead, length gal and base width
	gab, at zk.
drawnormalfourCA	drawnormalfourCA(i, j, k, 1) draws the normal thickness piecewise linear
	line zizjzkzl ending in a closed (black) arrowhead, length gal and base
	width gab, at zl.
drawnormalOA	drawnormalOA(i, j) draws the normal thickness line zizj, ending in an
	open arrowhead, length gal and base width gab, at zj.
drawnormalF	drawnormalF(i, j) draws the normal thickness line zizj, ending in a
	fanin, length gfl and base width gfb, at zj.
drawnormalFO	drawnormalFO(i, j) draws the normal thickness line zizj, ending in a
	fanin, length gfl and base width gfb, at zi and an open circle, diameter dotdiam,
	at zj.
drawthick	drawthick(i, j) draws the thick line zizj.
drawthick0	drawthickO(i, j) draws the thick line zizj, ending in an open circle,
	diameter dotdiam, at zj.
drawthick00	drawthick00(i, j) draws the thick line zizj, ending in open circles, di-
	ameter dotdiam, at zi and zj.



Figure 5: Calculated points on rectangular boxes (suffix is j and located at zjbl=zj)

smooth

smoothtwo

smooth(i, j, k) replaces the sharp corner on the line zi--zj--zk with an arc radius smoothrad.

smoothtwo(i, j, k, l) replaces the sharp corners on the line zi--zj--zk--zl
with arcs radius smoothrad.

smoothdash(i, j, k) replaces the sharp corner on the dashed line zi--zj--zk
with an arc radius smoothrad.

smoothdots(i, j, k) replaces the sharp corner on the dotted line zi--zj--zk
with an arc radius smoothrad.

2.2.4 Box drawing routines

Routines are provided for drawing the different kinds of EXPRESS-G boxes.

The first argument to the box drawing routines is the suffix of the point that will be the left-hand bottom corner point of the box. The last argument, except where noted below, is the text that is to be placed at the center of the box. Intermediate arguments, if any, are the length and height of the box.

Every box routine calculates the four corner points of the box and the midpoints of each side of the box. For example, if the suffix is j, then as illustrated in figure 5, the corner points will be called zjbl, zjbr, zjtr and zjtl and the midpoints will be called zjbm, zjmr, zjtm and zjml. The center point for postioning the text is zjc and the center of the text is placed at this point.

The text center point is not necessarily the geometric center of the box. The geometric center is not calculated but lies at the intersection of the lines zjml--zjmr and zjbm--zjtm.

A SCHEMA box is rectangular with a normal perimeter and the box is divided into top and bottom halves by a horizontal line.

drawSCHEMA(j, len, ht)(btex schema_a etex) draws a SCHEMA double rectangular box of length len and height ht with its left-hand bottom corner at zj. The string schema_a is placed at the text center of the box, which is central within the upper half.

drawBINARY

drawSCHEMA

A simple data type box is rectangular with a solid boundary and a solid interior vertical line near the right hand edge.

 ${\tt smoothdash}$

smoothdots

drawB00LEAN	<pre>drawBINARY(j) draws a simple data type rectangular box of length sdtbl and height sdtbh with its left-hand bottom corner at zj. The string BINARY is placed at the text center of the box, which is central within the main part of the box. drawBOOLEAN(j) draws a simple data type rectangular box of length sdtbl and height sdtbh with its left-hand bottom corner at zj. The string BOOLEAN is placed at the text center of the box, which is central within the main part of the</pre>
drawCOMPLEX	box. drawCOMPLEX(j) draws a simple data type rectangular box of length sdtbl and height sdtbh with its left-hand bottom corner at zj. The string COMPLEX is placed at the text center of the box, which is central within the main part of the
drawEXPRESSION	box. drawEXPRESSION(j) draws a simple data type rectangular box of length sdtbel and height sdtbeh with its left-hand bottom corner at zj. The string EXPRESSION is placed at the text center of the box, which is central within the
drawGENERIC	main part of the box. drawGENERIC(j) draws a simple data type rectangular box of length sdtbl and height sdtbh with its left-hand bottom corner at zj. The string GENERIC is placed at the text center of the box, which is central within the main part of the
drawINTEGER	box. drawINTEGER(j) draws a simple data type rectangular box of length sdtbl and height sdtbh with its left-hand bottom corner at zj. The string INTEGER is placed at the text center of the box, which is central within the main part of the
drawLOGICAL	box. drawLOGICAL(j) draws a simple data type rectangular box of length sdtbl and height sdtbh with its left-hand bottom corner at zj. The string LOGICAL is placed at the text center of the box, which is central within the main part of the box.
drawNUMBER	drawNUMBER(j) draws a simple data type rectangular box of length sdtbl and height sdtbh with its left-hand bottom corner at zj. The string NUMBER is placed at the text center of the box, which is central within the main part of the box.
drawREAL	drawREAL(j) draws a simple data type rectangular box of length sdtbl and height sdtbh with its left-hand bottom corner at zj. The string REAL is placed at the text center of the box, which is central within the main part of the box.
drawSTRING	drawSTRING(j) draws a simple data type rectangular box of length sdtbl and height sdtbh with its left-hand bottom corner at zj. The string STRING is placed at the text center of the box, which is central within the main part of the box.
drawENUM	An ENUMERATION box is a rectangular box with a dashed perimeter and an interior dashed vertical line near the right hand edge. drawENUM(j, len, ht)(btex an_enum etex) draws a rectangular dashed ENUMERATION type box of length len and height ht with its left-hand bot- tom corner at zj. The string an_enum is placed at the text center of the box,
drawSELECT	<pre>which is central within the main part of the box. A SELECT box is rectangular with a dashed boundary and an interior dashed vertical line near the left hand edge. drawSELECT(j, len, ht)(btex a_select etex) draws a rectangular dashed SELECT type box of length len and height ht with its left-hand bottom corner at</pre>

11

zj. The string a_select is placed at the text center of the box, which is central within the main part of the box.

drawTYPE(j, len, ht)(btex a_type etex) draws a rectangular dashed

TYPE type box of length len and height ht with its left-hand bottom corner at zj. The string a_type is placed at the text center of the box, which is central within the main part of the box. An ENTITY box is a rectangular box with a solid boundary. drawENT(j, len, ht)(btex an_entity etex) draws a rectangular EN-TITY type box of length len and height ht with its left-hand bottom corner at zj. The string an_entity is placed at the text center of the box, which is also the geometric center of the box. drawPREF A page reference box is a rectangle with rounded ends and a solid boundary. drawPREF(j, len, ht)(btex page ref etex) draws an oval page reference box of length len and height ht with its left-hand bottom corner at zj. The string page ref is placed at the text center of the box, which is also the geometric center of the box. An interschema box is a rectangle surrounding a page reference like box. The rectangular box may have either a solid or a dashed boundary.

A TYPE box is rectangular with a dashed boundary.

drawISU(j, len, ht)(btex sch.ent etex) draws a three-part rectangular interschema USE box of length len and height ht with its left-hand bottom corner at zj. The string sch.ent is placed at the text center of the box, which is also the geometric center of the box.

drawISUR(j, len, ht)(btex sch.ent etex)(btex newname etex) draws a three-part rectangular interschema USE and RENAME box of length len and height ht with its left-hand bottom corner at zj. The string sch.ent is placed at the text center of the box, which is also the geometric center of the box. The string **newname** is placed at the geometric center of the bottom third of the box (this is the point zjrnm).

drawISR(j, len, ht)(btex sch.ent etex) draws a three-part rectangular dashed interschema REFERENCE box of length len and height ht with its lefthand bottom corner at zj. The string sch.ent is placed at the text center of the box, which is also the geometric center of the box.

drawISRR(j, len, ht)(btex sch.ent etex)(btex newname etex) draws a three-part rectangular dashed interschema REFERENCE and RENAME box of length len and height ht with its left-hand bottom corner at zj. The string sch.ent is placed at the text center of the box, which is also the geometric center of the box. The string **newname** is placed at the geometric center of the bottom third of the box (this is the point zjrnm).

An EVENT box is a rhomboid with a solid boundary.

drawLEVENT(j, len, ht)(btex local_event etex) draws a rhomboidal Local EVENT box of length len, height ht and slope eventslope with its lefthand bottom corner at zj. The string local_event is placed at the text center of the box, which is also the geometric center of the box. Like the rectangular boxes, the calculated bottom and top middle points are vertically below and above the

drawENT

drawTYPE

drawISU

drawISUR

drawISR

drawISRR

drawLEVENT

12



Figure 6: Calculated points on rhomboidal boxes (suffix is j and located at zjbl=zj)

geometric center, but this means that they are *not* midway beteen the bottom and top corner points. This is illustrated in Figure 6.

drawGEVENT(j, len, ht)(btex global_event etex) draws a thick rhomboidal Global EVENT box of length len, height ht and slope eventslope with its left-hand bottom corner at zj. The string global_event is placed at the text center of the box, which is also the geometric center of the box.

drawcirclebox

drawdashcircle

drawGEVENT

A circle box is a circle with a solid boundary.

drawcirclebox(j, diam)(btex ABC etex) draws a circle diameter diam and center at zj. The string ABC is placed at the center of the circle. In this case the four midside points, zjbm, zjmr, zjtm and zjml, are at on the circle at the extreme bottom, right, top and left (i.e., at the S, E, N and W compass points). The corner points, zjbl, zjbr, zjtr and zjtl, are also on the circumference midway between the midpoints (i.e., at the SW, SE, NE and NW compass points).

drawdashcircle(j, diam) draws a dashed circle. The location and size parameters are the same as for drawcirclebox, as are the calculated points.

2.3 Extra BLA variables and routines

In addition to the facilities described above for EXPRESS-G diagrams, extra line styles and boxes are provided for other kinds of diagrams.

Additional line end styles are coded by: A for an Arrowhead, OD for an Open Diamond, CD for a Closed Diamond.

The variables gdl and gdb hold the the length and base width of diamond line end styles (see Figure 7), analagous to the corresponding variables for arrow heads and fanin. By default, gdl is twice the default arrowhead length and gdb is three quarters of the default arrowhead base width.

drawA drawA(i, j) draws a simple arrowhead, length gal and base width gab, at the zj end of the vector from zi to zj.

drawDCA(i, j) draws a double closed arrowhead, each of length gal and base width gab, at the zj end of the vector from zi to zj.

drawOD

drawDCA

gdl

gdb

drawOD(i, j) draws an open diamond, length gdl and base width gdb, at



Figure 7: Length parameters for diamond line end styles

the zj end of the vector from zi to zj.

drawCD(i, j) draws a closed (black) diamond, length gdl and base width gdb, at the zj end of the vector from zi to zj.

drawdashA(i, j) draws the dashed line zi--zj, ending in an arrowhead, length gal and base width gab, at zj.

drawdashOA(i, j) draws the dashed line zi--zj, ending in an open arrowhead, length gal and base width gab, at zj.

drawnormalOD(i, j) draws the normal thickness line zi--zj, ending in an open diamond, length gdl and base width gdb, at zj.

drawnormalCD(i, j) draws the normal thickness line zi--zj, ending in a closed (black) diamond, length gdl and base width gdb, at zj.

drawcircleA(j, diam) draws a circle diameter diam centered at zj. A counterclockwise pointing arrow is placed at the top of the circle. The midside and 'corner' points are the same as calculated by the drawcirclebox routine.

drawDot(j, diam) draws a black dot, diameter diam centered at zj. There are no calculated points.

drawCircledDot(j, diam) draws a black dot with a circle outside it, overall diameter diam, centered at zj. There are no calculated points. This could be used for a UML 'bullseye'. The white ring should obscure anything drawn earlier that lies underneath it (by definition the black dot covers up anything underneath itself). For example:

draw z1z2;	%	line from z1 to z2
<pre>drawCircledDot(2,</pre>	len); %	obscures last len of above line
draw z3z2;	%	not obscured

Commands of the form draw...box have a text argument which is printed at the center of the box. If an empty box is required the text argument should be an empty string (""). For example draw...box(...)("").

drawcardbox

drawcardbox(j, len, ht, fold)(btex note etex) draws a rectangular box, length len and height ht, with its bottom left corner at zj. The top right corner of the box is folded down, with the side length of the fold given by fold. The string note is placed at the center of the box. If no text is required, then use

drawdashOA drawnormalOD

drawCD

drawdashA

drawnormalCD

drawcircleA

drawDot

drawCircledDot



Figure 8: Calculated points on diamond boxes (suffix is j and located at zjc=zj)

an empty string (i.e., ""), as in \drawcardbox(23, el, eh, 1/8eh)("")

drawdiamondbox

drawdiamondbox(j, len, ht)(btex str etex) draws a diamond shaped box, length len and height ht, with its center at zj. The usual corner, center and midpoints are also calculated except that the points z\$tm etc., are at the corners of the diamond and the points z\$tl etc., are at the middle of each line, as shown in Figure 8. The text str is placed at the center of the box.

drawtwodiamondbox

drawtwodiamondbox(j, len, ht, inset)(btex str etex) draws a diamond shaped box, length len and height ht, with its center at zj. The text str is placed at the center of the box. A second diamond is drawn inside the first. The parameter inset is the space between adjacent outer and inner lines. No calculated points are available for the inner diamond.

drawdoublerectangle(j, len, ht, tf) draws a rectangular box, length len and height ht, located with its bottom left corner at zj. The box is divided horizontally into a top and bottom portion, with the height of the top portion being the fraction tf of the total height of the box. The center points of the two portions are calculated as zjct and zjcb for the top and bottom respectively. The ends of the dividing line are at zjtfl and zjtfr.

drawtriplerectangle

drawdoublerectangle

drawtriplerectangle(j, len, ht, tf, bf) draws a rectangular box, length len and height ht, located with its bottom left corner at zj. The box is divided horizontally into a top, middle and bottom portion. The height of the top portion is the fraction tf of the total height of the box. The height of the bottom portion is the fraction bf of the total height of the box. The center points of the three portions are calculated as zjct, zjcm and zjcb for the top, middle and bottom respectively. The ends of the dividing lines are at zjtfl and zjtfr for the upper line, and at zjbfl and zjbfr for the lower line.

hiderectangle hiderectangle(j, len, ht) draws an invisible rectangular box, length len and height ht, located with its bottom left corner at zj. The box covers up

drawdashboxover

and hence cannot be used for positioning or alignment purposes. r drawdashboxover(j, len, ht) draws a dashed rectangular box, length len

anything underneath it. Note that unlike all the other boxes the only point handle to the box is the bottom left corner — the zjbl, zjtr, etc., points are not available

and height ht, located with its bottom left corner at zj. The box covers up anything underneath it.

drawindexbox

drawovalbox

drawdashellipse

drawdrum

drawindexbox(j, len, ht, lenp, htp)(btex str etex) draws a rectangular box, length len and height ht, located with its bottom left corner at zj. Another rectangular box, length lenp and height htp, is drawn with its bottom left corner at the top left corner of the first box (like an index card). The main box points are zjbl etc., but the secondary box points are zjP.bl, zjP.bm etc. The text str is placed at the center (zjP.c) of the secondary box.

drawroundedbox (j, len, ht, rad)(btex str etex) draws a rectangular box, length len and height ht, located with its bottom left corner at zj. The corners of the box are rounded using radius rad. If 2rad is greater than len or ht, then 2rad is reduced to the lesser of len and ht, thereby producing at least one pair of semicircular sides. The text str is placed at the center of the box.

> drawovalbox(j, len, ht)(btex str etex) draws an elliptical box, horizontal diameter len and vertical diameter ht, located with its center at zj. Like the diamond box, zjtm, zjmr, etc., are at the extremes of the ellipse and the points zjtr, zjbr, etc., are on the ellipse approximately halfway between the extremum points, similar to the positions shown in Figure 8. The text str is placed at the center of the box.

drawoutputbox drawoutputbox(j, len, ht)(btex str etex) draws a box, length len and height ht, located with its bottom left corner at zj. The bottom of the box is a wavy line and the other three sides are straight. The text str is placed at the center of the box. This kind of box is often used as a flowchart symbol for output; it is meant to be a represention of printed paper which has been torn from a long continuous roll.

drawdashellipse(j, len, ht) draws a dashed ellipse. The location and dimensional parameters correspond to those for drawovalbox, as do the calculated points.

drawdrum(j, len, ht)(btex str etex) draws a 'drum' box. This is constructed as a rectangular box of width width len and height ht, with its bottom left hand corner at zj. The horizontal line at the bottom of the rectangle is replaced by a half-ellipse with a major diameter of len and the minor diameter given by drumlid*len. The horizontal line at the top is replaced by a full ellipse with the same dimensions.

As shown in Figure 9, the calculated points on the vertical sides of the drum are the same as for a rectangular box. There are three calculated points on the base of the drum, called zjbml, zjbm and zjbmr. Similarly there are three points on the top of the drum called zjtml, zjtm and zjtmr. The center point, zjc is halfway between the sides and halfway between the two lower ellipse halves. The text str is placed at this center point.

drawstickman dr

drawstickman(j, len, ht) draws a genderless full frontal (or rear) stick figure enclosed in an invisible box, length len and height ht, located with its bottom left corner at zj. This can be used, for example to represent an actor in a UML Use Case diagram. Setting the height to be twice the length produces a reasonable result, for instance: drawstickman(3, wd, 2wd);



Figure 9: Calculated points on a drum (suffix is j and located at zjbl=zj)

3 An example

The example attempts to bring out the main methods that can be used in defining EXPRESS-G diagrams. The code for eleven diagrams is provided, none of which necessarily has anything to do with a real example of EXPRESS-G modeling, but rather is meant to be illustrative of diagramming techniques.

This section is best studied with a copy of the printed diagrams to hand. Section 3.5 on page 55 explains briefly how to run the example and obtain the diagrams.

The default extension for a METAPOST file is .mp.

```
1 \langle *eg \rangle
2 %%% EXPEG.MP Example MetaPost EXPRESS-G diagrams
3
```

Assuming that the diagrams are to be eventually embedded in, or processed through, a IATEX document then any TEX macros must come at the start of the file between a verbatimtex ... etex pair. When dealing with text, METAPOST does not handle more than a single line. The macro below is to enable two lines of text to be handled by METAPOST.

```
4
5% use btex \twolines{first}{second} etex for two-line labels
6 verbatimtex \def\twolines#1#2{\vbox{\hbox{#1} \hbox{#2}}} etex
7
```

The expressg package file has to be input to make the routines available. 8 input expressg 9

17

3.1 Diagram 1: Some boxes and lines

Many diagrams can be defined in one file. Each diagram starts with beginfig(N), where N is a unique integer, and ends with endfig;. Encapsulated PostScript for each diagram is written to a file called mproot.N, where mproot is the root name of the input file. In our case the output files will be expeg.1 through expeg.11. 10 beginfig(1)

11

The first diagram consists of some of the expressg box and line styles. The main point is to illustrate how METAPOST allows the boxes to be aligned. Start off by defining a point z1 at the origin. By default, a variable name starting with the letter z is assumed to be a point. Similarly, variables starting with x and y are assumed to be x- and y-coordinates. If zk is defined with a value then automatically xk is the value of the x coordinate of the point zk and yk is the value of the x conversely, if xj and yj are defined, zj is automatically defined.

Note that the = sign denotes an equation, not an assignment. The power of METAPOST comes from its ability to solve linear equations.

Having defined the point z1, draw an EXPRESSION box with its bottom left hand corner at z1.

```
12 z1=(0,0);
13 drawEXPRESSION(1);
14
```

Remember that when a box is drawn, its corner and mid points are calculated. These can be used in defining other points. In particular there is the point z1tr which is the top right hand corner of the EXPRESSION box. Also, if another box is drawn at point z2 then it will have its bottom left hand corner at z2b1.

The point z2bl is defined by a linear equation with respect to the point z2. As METAPOST can solve linear equations, given the point z2bl it can solve for the point z2.

The next box has its bottom left hand corner at the top right hand corner of the EXPRESSION box, by specifying that the point z2bl and z1tr are the same location and then drawing a box located at z2. The third (BOOLEAN) box is also stacked in the same way with respect to the second (INTEGER) box.

```
15 z2bl=z1tr;
16 drawINTEGER(2);
17
18 z3bl=z2tr;
19 drawBOOLEAN(3);
20
```

Next a piece of circled text is drawn. Using the same method as before, the bottom of the circle is made to be coincident with the middle of the top of the BOOLEAN box. The namespace routine is used to specify the circle diameter. Note that an argument to a routine can itself be a routine.

```
21 z4bm=z3tm;
```

22 drawcirclebox(4, namespace(btex ABS etex)(nextra))(btex ABS etex); 23

METAPOST is a typed language, and new variables can be declared of a particular type. A numeric variable holds a number (or a length). The diameter of the circle just drawn is going to be used as a length in later parts of this diagram, so it is sensible to store its value in a new variable called **diam**. We then use it in drawing another circle directly on top of the first one.

```
24 numeric diam;
25 diam = namespace(btex ABS etex)(nextra);
26
27 z5bm=z4tm;
28 drawcirclebox(5, diam)(btex SAB etex);
29
```

The next element in the diagram is an ENTITY box. We use the default height for the entity box and the default page connector length as its length (no reason, but it saves inventing another variable or value).

The ENTITY box is positioned with respect to the SAB circle, but in a slightly more complicated way than earlier positions. We want the left hand side of the box to align with the right hand side of the circle. Also, the middle of the box should align with the top of the circle.

The ENTITY box will be located at point z6. The easiest way is to specify z6 is z6ml=(x5mr, y5tm), but to demonstrate more of METAPOST's equation solving abilities a more complicated approach is taken to the specification of the x coordinate. The horizontal distance between the points z5ml and z5tl on the circle is the same as the distance between the points z5tr and z5mr. We can express this as x5ml-x5tl = x5tr-x5mr. As we want x6ml, which is the same as x6bl, to be the same as x5mr we can substitute into this equation, as done in the code below. Note that METAPOST can solve for the unknown x6bl even though it is on the right hand side of the equation because the other three values are known.

The y coordinate of the point z6mr is specified as equal to the y coordinate of the top of the circle. Because of the linear relations involving all the points on a box, it is sufficient to provide one x coordinate and one y coordinate to fix all the points on the box.

```
30 x5ml-x5tl = x5tr-x6bl;
31 y6mr=y5tm;
32 drawENT(6, pconl, enth)(btex an\_ent etex);
33
```

The geometric center of the next box, an LEVENT, is vertically above the geometric center of the ENTITY box, and the bottom of LEVENT is slightly (one lenth unit) above the top of the ENTITY box. This is accomplished here by specifying z7bm in terms of expressions involving the coordinates of the point z6tm.

```
34 z7bm=(x6tm,y6tm+u);
35 drawLEVENT(7, pconl, eventh)(btex levent etex);
36
```

At point z8 a GEVENT box is drawn. It's bottom left hand corner is vertically above the top left hand corner of the LEVENT box. If you look at the resulting diagram, you will see that GEVENT is shifted right with respect to LEVENT. For vertical alignment of EVENT boxes it is best to use the middle rather than corner points.

```
37 z8=(x7t1, y7tr+u);
38 drawGEVENT(8, pcon1, eventh)(btex gevent etex);
39
```

mediation

For the last of the aligned boxes another aspect of METAPOST is used. META-POST provides a *mediation* facility whose syntax is frac[a,b]. The result of a mediation is a value that is frac between the value a and the value b. For example, the result of 0.25[10,30] is 15. Similarly, the result of 1/3[zj,zk] is the point one third of the way between point zj and point zk.

The SCHEMA box is positioned with its bottom right hand corner at one eighth of the way along, and at the top of, the BOOLEAN box.

```
40 z9br=1/8[z3t1,z3tr];
```

```
41 drawSCHEMA(9, pconl, schemah)(btex a\_schema etex);
42
```

The second half of the diagram illustrates various line and end styles. These are all horizontal, the same length and are aligned vertically, so define the x coordinate xmid of the left hand end of the lines, and the length len of the lines.

```
43 numeric xmid, len;
44 xmid = x8tr + diam;
45 len = 2diam;
46
```

shifted

The next line of code exhibits another METAPOST function. A point can be specified as being some other point that is *shifted* by given amounts in the x and y directions. More precisely, the shift is specified by a pair of values, written as (a,b). The first line is aligned vertically with the center of the first box and its start point is z11. The end point, z111, is specified as being the start point shifted horizontally by the length of the line.

```
47 z11=(xmid, y1ml); z111=z11 shifted (len, 0);
48 drawdashO(11, 111);
49
```

Because we will be using the same horizontal shift for all the lines, define a *pair*, called **moveright**, that is the required shift. The next two lines are also aligned vertically with the centers of the second and third boxes.

```
50 pair moveright;
51 moveright = (len, 0);
52
53 z12=(xmid, y2ml); z121=z12 shifted moveright;
54 drawnormalO(12, 121);
55
56 z13=(xmid, y3ml); z131=z13 shifted moveright;
57 drawthickO(13, 131);
58
```

The remaining lines will have the same vertical spacing as the first three, so define another pair, moveup, that we can use for vertical shift.

```
59 pair moveup;
60 moveup = (0, y13-y12);
61
62 z14=z13 shifted moveup; z141=z14 shifted moveright;
63 drawnormalD(14, 141);
64
65 z15=z14 shifted moveup; z151=z15 shifted moveright;
66 drawnormalCA(15, 151);
67
68 z16=z15 shifted moveup; z161=z16 shifted moveright;
69 drawnormalOA(16, 161);
70
71 z17=z16 shifted moveup; z171=z17 shifted moveright;
72 drawnormalF(17, 171);
73
```

The upper part of the diagram illustrates some of the additional line styles and boxes provided that are not part of the EXPRESS-G language.

The upper half is seperated from the bottom half by a dashed line, just to make the distinction. The line is set above the highest box so far, which is located at **z8**, and extends across the width of the diagram.

```
74 z300=(0,y8tm+diam); z301=(x111,y300);
75 draw z300--z301 dashes;
76
```

Define mup to be a distance, which we will use for vertical seperation.

```
77 numeric mup; mup := onelineh;
78
```

As previously, we'll start with the boxes which will be placed in a column at the left side of the diagram. The first box, high enough for one line of enclosed text, has its top right hand corner folded down by one quarter the box height. Text is placed at the center of the box, but if this is not required then it can be given as an empty string (i.e., "").

```
79 z302=(0,y300+mup);
80 drawcardbox(302, pconl, onelineh, 1/4onelineh)("");
81
```

The next box is a diamond shape. Unlike the rectangular and rhomboidal boxes, these are positioned via the center of the box instead of the bottom left hand corner.

```
82 z303bm=(x302tm, y302tm+mup);
83 drawdiamondbox(303, pconl, 2onelineh)(btex jewel etex);
84
```

Now there is a double box with the top portion being 3/4 of the box height. Text is put into the top half of the box using METAPOST's labeling routine (see page 26).

```
85 z304=(0, y303tm+mup);
86 drawdoublerectangle(304, pconl, 2onelineh, 3/4);
87 label(btex top etex, z304ct);
88
```

Next comes a triple box with the bottom portion being one fifth of the height, and the top portion being two fifths of the height. Text is placed in the top and middle portions, but not into the bottom.

```
89 z305=(0, y304tm+mup);
90 drawtriplerectangle(305, pconl, 3onelineh, 2/5, 0.2);
91 label(btex top etex, z305ct);
92 label("middle", z305cm);
93
```

Some languages include a box that looks a little like an index card, and this is shown next. A small box is also drawn inside the main part of the index box, just to illustrate that drawings may be placed inside boxes.

```
94 z306=(0,y305tm+mup);
95 drawindexbox(306, 2pconl, 3onelineh, 3/4pconl, onelineh)(btex pack etex);
96 z307=1/8[z306bl,z306tr];
97 drawENT(307, 1/4pconl, onelineh)(btex E etex);
98
```

Another box shown here is a dotted box that covers up anything that is underneath it. To demonstrate, a folded corner box is drawn first, and then the dashed box is drawn second. They have to be done in this order. (The reason for the notation y306P.tl is explained under diagram 2 on page 27).

```
99 z308=(0, y306P.tl+mup);
100 z309ml=z308c;
101 drawcardbox(308, pconl, 2onelineh, 1/4onelineh)(btex covered etex);
102 drawdashboxover(309, pconl, onelineh);
103
```

An oval box comes next. The perimeter is an ellipse.

```
104 z310bm=(x306bm,y309tm+mup);
```

```
105 drawovalbox(310, 2pconl, 2onelineh)("");
106
```

Now we have two rounded boxes. In the first one the corner radius is small and the corners are neatly rounded. The radius is large in the second case, and it is automatically reduced to give semicircular ends. Further, the center portion of the second of these boxes is 'deleted' by putting a hiderectangle over it.

```
107 z311ml=(x310mr+mup, y310ml);
108 drawroundedbox(311, 2pconl, 2onelineh, 1/2onelineh)("rounded");
109
110 z312ml=(x311mr+mup, y311mr);
111 drawroundedbox(312, 2pconl, 2onelineh, 3/2onelineh)(btex large radius etex);
112
113 z314=(x312bm-1/4pconl, y312bm-1/2mup);
114 hiderectangle(314, 1/2pconl, (3/2onelineh + mup));
115
```

The last of the simple enclosed shapes are a dashed ellipse and a dashed circle, which are aligned below the partially obscured rounded box.

```
116 z315tm=(x312bm, y312bm-mup);
117 drawdashellipse(315, 2pconl, 2onelineh);
118
119 z316tm=(x315bm, y315bm-mup);
120 drawdashcircle(316, diam);
121
```

A drum box is now presented. This is centered with respect to the rounded box and aligned with the bottom of the triple box. It is the same nominal size as the triple box.

```
122 z317bc=(x311bm,y304br);
```

123 drawdrum(317, pconl, 3onelineh)(btex drum etex);

To be a little different, I'll add a 'handle' to the top of the drum. The top of the handle will be above the drum at the point z317A, and the attachment points are at the tml and tmr points on the drum. The normally sharp join in the middle of the handle is smoothed off.

```
124 z317A=(x317tc, y317tc+2onelineh);
125 drawnormalthree(317tml, 317A, 317tmr);
126 smooth(317tml, 317A, 317tmr);
127
```

Below the drum box is a stick figure. It is centered with respect to the drum box and based on the bottom row.

```
128 z320bm = (x317bm, y302);
129 drawstickman(320, onelineh, 2onelineh);
130
```

Above the drum box there is an output box, vertically aligned with the stick figure, labelled 'output'.

```
131 z325bm = (x320bm, y306);
132 drawoutputbox(325, pconl, 2onelineh)(btex output etex);
133
```

Above the output box there is a simple circled dot.

```
134 z330 = (x320bm, y309ml);
135 drawCircledDot(330, 3/2onelineh);
136
```

As in the bottom half, line styles are drawn on the right side of the diagram. First, though, there is an arrowed circle symbol. This is vertically aligned with the centers of the lines.

```
137 z401bm=(1/2[x11,x111], y301+mup);
138 drawcircleA(401, diam);
139
```

Finish off with assorted linestyles.

```
140 z402=(xmid, y401tm+mup); z502=(x111, y402);
141 drawdashA(402, 502);
```

```
142
143 z403=(xmid, y402+mup); z503=(x111, y403);
144 drawdashOA(403, 503);
145
146 z404=(xmid, y403+mup); z504=(x111, y404);
147 drawnormalOD(404, 504);
148
149 z405=(xmid, y404+mup); z505=(x111, y405);
150 drawnormalCD(405, 505);
151
152 z406=(xmid, y405+mup); z506=(x111, y406);
153 drawnormalDCA(406, 506);
154
```

Lines can be drawn at any angle, and the non-circular line end styles remain aligned.

rotatedaround

A set of lines and endstyles are drawn rotating about the common start point of the lines. pt1 rotatatedaround(pt2, ang) is METAPOST code that rotates the point pt1 around the point pt2 counterclockwise through the angle ang in degrees.

```
155 numeric ang; ang = 180/7;
156 z600=(xmid, y406+2mup);
157 z699=(x600+len, y600);
158 z601=z699 rotatedaround(z600, ang);
159 z602=z699 rotatedaround(z600, 2ang);
160 z603=z699 rotatedaround(z600, 3ang);
161 z604=z699 rotatedaround(z600, 4ang);
162 z605=z699 rotatedaround(z600, 5ang);
163 z606=z699 rotatedaround(z600, 6ang);
164 z607=z699 rotatedaround(z600, 7ang);
165 \, \mathrm{drawnormalCA(600,601)};
166 drawnormalOA(600,602);
167 drawnormalF(600,603);
168 drawdashA(600,604);
169 drawnormalOD(600,605);
170 drawnormalCD(600,606);
171
```

This is the end of the first diagram. 172 endfig; % end figure 1 173

3.2 Diagrams 2–5: EXPRESS-G like models

The diagrams in this section are all drawn using the EXPRESS-G specific routines. Nevertheless, they do illustrate general BLA techniques.

3.2.1 Diagram 2: Schema level diagram

In contrast to the first diagram, this one is much more like an EXPRESS-G diagram. It illustrates a diagram consisting of SCHEMA boxes and interconnections.

174 beginfig(2)% schema level model175

First define some lengths to assist in laying out the diagram, which is going to consists of three rows of boxes with varying numbers of boxes in each row; the bottom row has two, the middle has four, and the top row has one. We also want to align the boxes vertically in a 'nice' manner.

```
176 numeric lb, hb; % length & height of boxes
177 lb=20u; hb=schemah;
178 numeric upshift; % vertical distance between rows
179 upshift = 20u;
180 numeric sideshift; % horizontal distance between boxes
181 sideshift = 10u;
182
```

The bottom row has two boxes and the second four. The two bottom boxes will be aligned vertically under the two middle boxes in the middle row. Allowance must be made for a 'missing' box at the start of the bottom row. The second box on the bottom row is spaced from the first one.

```
183 %%% bottom row
184
185 z9=(lb+sideshift,0);
186 drawSCHEMA(9, lb, hb)("nine");
187
188 z11=(x9br+sideshift,y9);
189 drawSCHEMA(11, lb, hb)("eleven");
190
```

The bottom of the boxes on the second row are all at a fixed distance above the top of the boxes on the first row. We only need to specify the y coordinate of the bottom of one box in the row, and use that for the remainder,

```
191 %%% middle row
192
193 z1=(0,y9tl+upshift);
194 drawSCHEMA(1, lb, hb)("one");
195
196 z3=(x1br+sideshift, y1);
197 drawSCHEMA(3, lb, hb)("three");
198
199 z5=(x3br+sideshift, y1);
200 drawSCHEMA(5, lb, hb)("five");
201
202 z7=(x5br+sideshift, y1);
203 drawSCHEMA(7, lb, hb)("seven");
204
```

The single box on the top row will be position centrally above the two middle boxes of the middle row. Mediation is used to calculate the x coordinate that is halfway between the right side of the left box and the left side of the right box.

```
205 %%% top row
206
207 x50 = 1/2[x3br,x5b1];
208 z31bm=(x50,y1tl+upshift);
209 drawSCHEMA(31, lb, hb)("thirtyone");
210
```

This completes the boxes. The diagram is finished off by drawing connections between the boxes. The first set of connections are just straight lines between known points on boxes.

```
211 %%% connectors
212
213 drawdashO0(9tm, 3bm); % three/nine
214 drawdashO(5bm, 11tm); % five/eleven
215 drawdashO(3ml, 1mr); % three/one
216 drawdashO(3mr, 5ml); % three/five
217 drawdashO0(9mr, 11ml); % nine/eleven
218
```

The next connection is more complex. It goes from schema thirtyone to schema nine on a three-legged route. It goes horizontally left from thirtynine, then vertically down, and finally horizontally right to nine. It also passes to the left of schema one.

The route is constructed by specifying a point **z90** to the left of schema nine, and the point **z31A** on the left hand of schema thirtynine. The VH routine is then called to generate the corner point on the (rotated) L shaped path between **z90** and **z31A**. The connection is then drawn using these four points.

```
219 %%% thirtyone/nine
220 z90=((x1bl-sideshift), y9ml);
221 z31A=3/4[z31bl,z31tl];
222 VH(90, 31A);
223 drawdash0(90vh, 31A);
224 drawdashthree0(90vh, 90, 9ml);
```

pickup

To demonstrate another aspect of METAPOST, a dot is drawn at the location of z90. If the drawing pen is thick enough, just drawing one point is sufficient to make a noticeable dot. The code pickup thickpen; drops the current pen and picks up the thickpen for drawing. The code pickup normalpen; picks up the normal pen, which is the one that is normally used.

label dotlabel The last line of code here is label.lft("z90", z90); which puts the text 'z90' at the left of point z90. The label routine has two arguments, the first the text and the second the point at or near which the text is to be placed. The simplest form of the routine is label("text", zk) which centers the text over the point zk. A suffix can be used after the label (e.g., .lft in the code below) to modify the placement. .lft positions the text at the left of the point. The other modifiers are: .rt, .top, .bot, .ulft, .urt, .llft and .lrt. There is also a dotlabel.suffix("text", zk) routine which draws the point as well. For complicated diagrams it can be useful to label some points during development, and then comment them out for the final picture.

```
225 pickup thickpen;
226 draw z90;
227 pickup normalpen;
228 label.lft("z90", z90);
229
```

The next two connectors, between schemas one and eleven and between thirtyone and eleven are of a similar U shape. There is, though, one tricky point to remember. Up till now, all point arguments to the expressg routines have been of the form 3 or 4bm; that is, either a number or a number followed by letter(s). For example, VH(3,4bm) will calculate the point z3vh. However, if a routine adds more letters to an argument that already has letters, such as VH(4bm,3), the calculated point is *not* z4bmvh as might be expected, but instead is z4bm.vh. This has tripped me up on several occasions.

```
230 %%% one/eleven
231 z1A=1/3[z1bl,z1br];
232 z11A=(x11bm, (y11bm-sideshift));
233 VH(1A, 11A);
234 drawdashthree(1A, 1A.vh, 11A); % note it is 1A.vh and not 1Avh
235 drawdash0(11A, 11bm);
236 dotlabel.rt("z11A", z11A);
237 dotlabel.lft("z1A.vh", z1A.vh); % note it is z1A.vh and not z1Avh
238
```

The connector between thirtyone and eleven is the same general shape as the thirtyone to nine connector. This time, I'll use the HxH routine to calculate the turning points

```
239 %%% thirtyone/eleven
240 z31B=3/4[z31br,z31tr];
241 x31BC=x7br+sideshift;
242 HxH(31B, 31BC, 11mr);
243 drawdashfourO(31B, 31B.hxh, 11mr.hxh, 11mr);
244
```

The next two connectors, between thirtyone and seven, and between thirtyone and one, are simple L shapes, which are easier to produce than the U shaped ones.

```
245 %%% thirtyone/seven
246 z31D=1/4[z31br,z31tr];
247 VH(7tm, 31D);
248 drawnormalthree0(31D, 7tm.vh, 7tm);
249
250 %%% thirtyone/one
251 z31C=1/4[z31b1,z31t1];
252 VH(1tm, 31C);
253 drawdash0(1tm.vh, 31C);
255
```

The last two connectors are also simple, each consisting of vertical then horizontal then vertical lines and the turning points can be calculated by the VhVroutine..

```
256 %%% thirtyone/three
257 z31E=1/4[z31bl,z31br];
258 VhV(31E, 3tm);
259 drawnormalfourO(31E, 31E.vhv, 3tm.vhv, 3tm);
260
261 %%% thirtyone/five
262 z31F=3/4[z31bl,z31br];
263 VhV(31F, 5tm);
264 drawnormalfourO(31F, 31F.vhv, 5tm.vhv, 5tm);
265
```

The last element in the diagram is a vertical closed arrowheaded line with a label alongside. The arrowhead is on the top line of the connector between thirtyone and eleven. Note the point mediation expression expression in the **label** routine call.

```
266 %%% labeled arrow
267 z3111=1/4[z31B,z31B.hxh];
268 numeric arrowsize;
269 arrowsize := schemah;
270 z3112=(x3111, y3111+arrowsize);
271 drawnormalCA(3112, 3111);
272 label.rt(btex an\_import etex, 1/2[z3111, z3112]);
273
```

The end of this diagram.

```
274 endfig; % end fig 2
275
```

3.2.2 Diagram 3: Tree structure

This diagram is the first of three that are related in style. It is useful at this point to define some variables, and values for some of these as they will probably be used in all three diagrams.

Values are assigned (e.g., var := val;) so that they can be changed later if necessary (e.g., var := newval;). If values were equated like var = val; then they cannot be changed afterwards³ as var = newval; will then have specified two incompatible equations for var and METAPOSTwill report a problem.

```
276
277 %/%% some commonly used variables & values
278
279 numeric pl, ph; % length & height of numeric page connectors
280 pl:=pconl; ph:=pconh; % for eg., 9,9 (9,9)
281 numeric plnamed; % length of named page connector
282 numeric irh; % height of interschema boxes
```

³Except by assignment.

```
283 irh:=ish;
                              % normally one line of text
284 numeric sdtl, sdth;
                              % length & height of SDT (e.g. INTEGER) boxes
285 sdtl:=sdtbl; sdth:=sdtbh;
                              % length & height of enum boxes (inset is sdtbs)
286 numeric edtl, edth;
287 edth:=sdtbh;
                              % normally one text line
288 numeric el, eh;
                              % length & height of entity boxes
289 eh:=enth;
                              % normally one text line
                              % vertical space between rows of boxes
290 numeric upshift;
291 numeric nextup;
                              % vertical space between bases of rows of boxes
```

Now another bit of METAPOST. There is another type of variable called a **picture** variable. This can hold something as trivial as a dot or a text string, up to a complete diagram. References to an Index type is a common element in some of the diagrams. It is useful to create a shorthand for these by using picture variables.

```
292 picture sindex, pindex;
293 sindex := btex eleven.Index etex;  % interschema reference text
294 pindex := btex 9,9 Index etex;  % page reference text
295 numeric slindex, plindex;
296 slindex := namespace(sindex)(nextra);  % length of Index interschema
297 plindex := namespace(pindex)(nextra);  % length of Index page reference
```

Yet more new METAPOST. Array variables are supported. These are declared by giving the variable name immediately followed by a pair of square brackets. An element of an array can be simply accessed by suffixing the variable name with the integer number denoting the desired position. Actually we have been using array variables all along, as z3 or x7 are elements of the arrays z[] and x[] respectively. We define two arrays, one to be used for specifying horizontal spaces (lengths) and the other for vertical (y) coordinates.

```
298 numeric hspace[];% horizontal spaces299 numeric lyc[];% Y coords of box bases300301 %/%/%/%/%/% end commonly used variables & values302
```

Diagram 3 consists of type boxes. We will make these all the same length and height.

```
      303 beginfig(3)
      % example fig 3

      304 %%%drawgrid;
      305
```

The space required for the longest name is edtl.

306 edtl := namespace(btex standard_data_name etex)(niextra);

There are several rows in the diagram. Specify the vertical space between the rows, and the space between the bottoms of the boxes (the height of a box is the normal height, sdth, for a type box). Also specify the y coordinates for each row, counting from the bottom.

```
307 upshift=2sdth;
308 nextup=sdth+upshift;
```

```
309 %%% the y coords of box bases
310 lyc1:=0;
311 lyc2:=lyc1+nextup;
312 lyc3:=lyc2+nextup;
313 lyc4:=lyc3+nextup;
314 lyc5:=lyc4+nextup;
315 lyc6:=lyc5+nextup;
316 lyc7:=lyc6+nextup;
317 lyc8:=lyc7+nextup;
Specify two horizontal spaces.
318 hspace1 := 1/3edt1; % space between data enums
319 hspace2 := 2/3edt1; % initial space at start of second row
320
```

Draw the 3 boxes in the first row and the 3 in the second; these are staggered with respect to each other and overlap as otherwise there there is not enough space to get them all onto one page. I'll use a numbering scheme for points so that zCR is a point in the Cth column and the Rth row.

```
321 %%% bottom two rows (1 & 2)
322
323 z11=(0,lyc1);
324 drawENUM(11, edtl, edth)(btex coordinates etex);
325
326 z22=(hspace2, lyc2);
327 drawENUM(22, edtl, edth)(btex solution\_data etex);
328
329 z31=(x11br+hspace1, lyc1);
330 drawENUM(31, edtl, edth)(btex pressure\_loads etex);
331
332 z42=(x22br+hspace1, lyc2);
333 drawENUM(42, edtl, edth)(btex nondimensional etex);
334
335 z51=(x31br+hspace1, lyc1);
336 drawENUM(51, edtl, edth)(btex temperature etex);
337
338 z62=(x42br+hspace1, lyc2);
339 drawENUM(62, edtl, edth)(btex force\_loads etex);
340
```

The third 'row' consists of a horizontal line stretching from the middle of the leftmost box to the middle of the rightmost box. There are connections from this line to the top middle of each of the boxes. We can use the VyV routine for the end points of the line.

```
341 %%% third row (horizontal line between 2 & 4 row)
342
343 z13=(x11tm,lyc3);
344 VyV(11tm, 13, 62tm);
345 drawnormal(11tm.vyv, 62tm.vyv);
```

Calculate the points on this line vertically above the midpoints of the boxes and draw the connections.

```
346 z23=(x22tm,lyc3); z33=(x31tm,lyc3); z43=(x42tm,lyc3); z53=(x51tm,lyc3);
347 drawnormal0(11tm.vyv,11tm);
348 drawnormal0(23,22tm);
349 drawnormal0(33,31tm);
350 drawnormal0(43,42tm);
351 drawnormal0(53,51tm);
352 drawnormal0(62tm.vyv,62tm);
353
```

The fourth row consists of three boxes. There is a vertical line from the bottom of the leftmost box to the horizontal line drawn in row 4. The three boxes are aligned horizontally with respect to this line.

```
354 %%% fourth row
355
356 z3224=1/3[z13,z62tm.vyv];
357 z3554=2/3[z13,z62tm.vyv];
358
359 %%% standard_data_name
360 z24bm=(x3224,lyc4);
361 drawSELECT(24, edtl, edth)(btex standard\_data\_name etex);
362 drawnormal (24bm, 3224);
363
364 %%% adhoc_data_name
365 z54bm=(x3554,lyc4);
366 drawTYPE(54, edtl, edth)(btex adhoc\_data\_name etex);
367
368 %%% STRING
369 z64br=(x62br,lyc4);
370 drawSTRING(64);
371
```

The fifth 'row' consists of two vertical connectors to the tops of the SELECT and TYPE boxes, with a horizontal line connecting the tops of the connectors.

```
372 %%% fifth row
373
374 z35=(1/2[x24tm,x54tm], lyc5);
375 VyV(24tm, 35, 54tm);
376 drawnormalthree0(35, 24tm.vyv, 24tm);
377 drawnormalthree0(35, 54tm.vyv, 54tm);
378
```

The top row consists of a SELECT box vertically aligned above the center of the horizontal line from the 5th row, together with a line joining these two, and also a page reference box off towards the right hand side

```
379 %%% sixth row
380
381 z36bm=(x35,lyc6);
382 drawSELECT(36,edtl, edth)(btex data\_name etex);
```

```
383 drawnormal(36bm,35);
384
385 %%% data_name
386 z66=(x62,lyc6);
387 drawPREF(66, sdtl, ph)(btex 3,1 (4) etex);
388
```

The remaining elements on the diagram are: a connector on the 4th row from adhoc_data_name to STRING; and a connector on the 6th row from the page reference to data_name.

```
389 %%%% rest of the connectors
390
391 %%% adhoc/STRING (54, 64)
392 drawnormalO(54mr,64ml);
393
394 %%% PREF/data_name (36, 66)
395 drawdashO(66ml,36mr);
396
The end of this diagram
397 endfig; % end fig 3
398
```

399

3.2.3 Diagram 4: Supertypes and subtypes

This diagram consists of a supertype entity and its four subtypes. The entities have various attributes.

```
% example 4
400 beginfig(4)
401 %%%drawgrid;
402
    Define some useful lengths.
403 %%% length of named page reference boxes
404 plnamed := namespace(btex 9,9 data\_name etex)(nextra);
405
406 %%% length of entity boxes
407 el := namespace(btex (ABS) Data\_t etex)(nextra);
408
409 upshift := 2sdth;
                              % vertical space between boxes
410 nextup := sdth+upshift;
                              % vertical space between bases of boxes
411 hspace1 := 1/2el;
                              % space between entity boxes
412
413 %%% the y coords of box bases
414 lyc1:=0;
415 lyc2:=lyc1+nextup;
416 lyc3:=lyc2+nextup;
417 lyc4:=lyc3+nextup;
418 lyc5:=lyc4+nextup;
419 lyc6:=lyc5+nextup;
```

```
420 lyc7:=lyc6+nextup;
421 lyc8:=lyc7+nextup;
422
```

We'll start with the second row which consists of 4 entity (subtype) boxes. The numbering scheme for point suffixes is zCR, where C is the column number and R is the row number.

```
423 %%%% 2nd row
424
425 %%% IntArray
426 z12=(0,lyc2);
427 drawENT(12, el, eh)(btex IntArray etex);
428
429 %%% RealArray
430 z22=(x12br+hspace1, lyc2);
431 drawENT(22, el, eh)(btex RealArray etex);
432
433 %%% CharArray
434 z42=(x22br+hspace1, lyc2);
435 drawENT(42, el, eh)(btex CharArray etex);
436
437 %%% BitArray
438 z52=(x42br+hspace1, lyc2);
439 drawENT(52, el, eh)(btex BitArray etex);
440
```

The first row is four simple data type boxes, aligned underneath the entity boxes.

```
441 %%%% 1st row
442
443 z11bm=(x12bm,lyc1);
444 drawINTEGER(11);
445
446 z21bm=(x22bm,lyc1);
447 drawREAL(21);
448
449 z41bm=(x42bm,lyc1);
450 drawSTRING(41);
451
452 z51bm=(x52bm,lyc1);
453 drawBINARY(51);
454
    The third 'row' is the thick lines connecting the subtype boxes.
455 %%% 3rd row
456
457 y13=
              y23=
                          y43=
                                      y53=1yc3;
458 x13=x12tm; x23=x22tm; x43=x42tm; x53=x52tm;
459 drawthick(13,53);
                           % horizontal line
460 %%% vertical connectors
461 drawthickO(13,12tm); drawthickO(23,22tm);
```

```
462 drawthickO(43, 42tm); drawthickO(53, 52tm);
463
464 z33=1/2[z13,z53]; % midpoint of horizontal line
465
```

The 4th row is an abstract supertype entity box, which is made 3 times taller than the default entity box height. The entity box is centered with respect to the row of subtype boxes and is connected to the subtype boxes connector. The digit 1 is placed near the connection point. There is also a page reference and an INTEGER box at the left and right (as attributes of the supertype).

```
466 %%% 4th row
467
468 %%% Data_t
469 z34bm=(x33,lyc4);
470 drawENT(34, el, 3eh)(btex \twolines{(ABS)}{Data\_t} etex);
471 drawthick(34bm,33);
472 label.urt("1", z33);
                           % label the connection point
473
474 %%% Index
475 z14=(0,lyc4);
476 drawPREF(14, plindex, ph)(pindex);
477
478 %%% INTEGER
479 z54br=(x51br,lyc4);
480 drawINTEGER(54);
481
```

The 5th row has page reference boxes at the left and right (as attributes of the supertype).

```
482 %%% 5th row
483
484 lyc5:=y34tm-ph; % change the initial y coord value
485
486 %%% Index
487 z15=(0,lyc5);
488 drawPREF(15, plindex, ph)(pindex);
489
490 %%% DataType
491 z55=(x54,lyc5);
492 drawPREF(55, plnamed, ph)(btex 3,1 data\_name etex);
493
```

The 6th, and last, row, just consists of a page reference centered above the supertype box.

```
494 %/% 6th row
495
496 %/% PREF to Data_t
497 lyc6:=lyc5+nextup;
498 z36bm=(x34tm,lyc6);
499 drawPREF(36, pl, ph)(btex 7,5 (6) etex);
500
```

```
The rest of the diagram consists of the labelled attribute lines.
501 %%%% attributes
502
503 %%% Sizes Data_t/Index (34, 14)
504 z3414=(x34,y14mr);
505 drawnormal(3414,14mr);
506 label.ulft(btex Sizes A[1:Limit] etex, z3414);
507
508 %%% num Data_t/INTEGER (34, 54)
509 z3454=(x34mr,y54ml);
510 drawnormal0(3454,54ml);
511 label.urt(btex (DER) num etex, z3454);
512
513 %%% Limit Data_t/Index (34, 15)
514 z3415=(x34,y15mr);
515 drawnormal(3415,15mr);
516 label.ulft(btex Limit etex, z3415);
517
518 %%% Kind Data_t/data_name (34, 55)
519 z3455=(x34mr, y55ml);
520 drawnormal(3455, 55ml);
521 label.urt(btex Kind etex, z3455);
522
523 %%% PREF into Data_t (36, 34)
524 drawdash0(36bm, 34tm);
525
526 %%% Data IntArray/INTEGER (12, 11)
527 drawnormalO(12bm, 11tm);
528 label.lrt(btex Data A[1:num] etex, z12bm);
529
530 %%% Data RealArray/REAL (22, 21)
531 drawnormalO(22bm, 21tm);
532 label.lrt(btex Data A[1:num] etex, z22bm);
533
534 %%% Data CharArray/STRING (42, 41)
535 drawnormalO(42bm, 41tm);
536 label.lrt(btex Data A[1:num] etex, z42bm);
537
538 %%% Data BitArray/BINARY (52, 51)
539 drawnormalO(52bm, 51tm);
540 label.lrt(btex Data A[1:num] etex, z52bm);
541
    The end of this diagram.
542 endfig;
                                          % end fig 4
543
```

3.2.4 Diagram 5: Portion of larger model

This diagram is more complex than the previous ones. It is, nevertheless, presented with virtually no commentary. It consists of three entity boxes and various page reference and interschema boxes to support the entities' attributes.

```
544 beginfig(5)
                                % example 5
545 %%drawgrid;
546
    As usual, specify commonly used variables and values.
547 plnamed := namespace(btex 7,4 Unstructured\_Donor etex)(nextra);
548
                                     % length of short and long ISR boxes
549 numeric irls, irll;
550 irls = namespace(btex eleven.MeshLocation etex)(nextra);
551 irll = namespace(btex eleven.MeshConnectivityType etex)(nextra);
552
553 upshift := irh;
554 nextup := irh+upshift;
555
                           \% max x coordinate value of LH of right column boxes
556 numeric maxrx;
557 maxrx := maxx-irrl;
                           % x coord of figure center
558 numeric xmid;
559 xmid := 1/2[irls, maxx-irll];
560
561 el := namespace(btex MeshConnectivity1\_1 etex)(nextra);
562 eh := irh;
563
564 %%% y coords
565 lyc1 := 0;
566 lyc2 := lyc1+nextup;
567 lyc3 := lyc2+nextup;
568 lyc4 := lyc3+nextup+irh;
569 lyc5 := lyc4+nextup;
570 lyc6 := lyc5+nextup;
571 lyc7 := lyc6+nextup;
572 lyc8 := lyc7+nextup;
573 lyc9 := lyc8+nextup;
574
    Rows one and two have one centered box each.
575 %%% Zcol/row
576 %%% row 1
577
578 %%% PREF for Overlapping
579 z21bm=(xmid,lyc1);
580 drawPREF(21, pl, ph)(btex 5,3 (8) etex)
581
582 %%% row 2
583
```

```
584 %%% OverlappingArea
```
```
585 z22bm=(x21bm,lyc2);
586 drawENT(22, el, eh)(btex OverlappingArea etex);
587
    Row three has three boxes, and the middle one is centered.
588 %%% row 3
589
590 %%% IndexArray
591 z13=(pl,lyc3);
592 drawISR(13, irls, irh) (btex eleven.IndexArray etex);
593
594 %%% Index
595 z23bm=(x22bm,lyc3);
596 drawISR(23, slindex, irh)(sindex);
597
598 %%% MeshLocation
599 z33br=(maxx,lyc3);
600 drawISR(33, irls, irh) (btex eleven.MeshLocation etex);
601
    Row four has three boxes, and the middle one is centered.
602 %%% row 4
603
604 %%% MeshConnectivity
605 z24bm=(x23bm,lyc4);
606 drawENT(24, el, eh)(btex MeshConnectivity etex);
607
608 %%% PREF for MeshConnectivity
609 z14ml=(pl,y24ml);
610 drawPREF(14, pl, ph)(btex 5,2 (8) etex);
611
612 %%% Structured_Donor
613 z34br=(maxx,lyc4);
614 drawPREF(34, plnamed, ph)(btex 7,5 Structured\_Donor etex);
615
    Rows five and six have one box each, at the right hand side.
616 %%% row 5
617
618 %%% Unstructured_donor
619 z35br=(maxx,lyc5);
620 drawPREF(35, plnamed, ph)(btex 7,4 Unstructured\_Donor etex);
621
622 %%% row 6
623
624 %%% MeshConnType
625 z36br=(maxx,lyc6);
626 drawISR(36, irll, irh) (btex eleven.MeshConnectivityType etex);
627
    Row seven has three boxes, and the middle one is centered.
```

```
628 %%% row 7
629
630 %%% IndexRange
631 z17=(0,lyc7);
632 drawISR(17, irls, irh)(btex eleven.IndexRange etex);
633
634 %%% INTEGER
635 x27bm=x24bm; y27ml=y17mr;
636 drawINTEGER(27);
637
638 %%% Zone
639 z37br=(maxx,lyc7);
640 drawISR(37, irls, irh) (btex seven.Zone etex);
641
    Rows eight and nine have one centered box each.
642 %%% row 8
643
644 %%% 1to1
645 z28bm=(x24bm,lyc8);
646 drawENT(28, el, eh)(btex MeshConnectivity1\_1 etex);
647
648 %%% row 9
649
650 %%% PREF to 1to1
651 z29bm=(x28bm,lyc9);
652 drawPREF(29, pl, ph)(btex 5,1 (8) etex);
653
```

All the boxes have been drawn. The remainder is adding and labelling attribute lines. First page references to subtype entities.

```
654 %%%% attributes, etc
655
656 drawthickO(21tm,22bm); % PREF to Overlap (21, 22)
657 drawthickO(14mr,24ml); % PREF to MeshConn (14, 24)
658 drawthickO(29bm,28tm); % PREF to 1to1 (29, 28)
659
```

The attributes of entity OverlappingArea (z22)

```
660 %%% PointListSize (22, 23)
661 drawdash0(22tm,23bm);
662 label.llft(btex PointListSize etex, z23bm);
663
664 %%% PointRange (22, 17)
665 z22A=1/4[z22b1,z22t1]; z17A=(1/2[0,x13], y17);
666 VH(17A,22A);
667 drawdashthree0(22A, 17A.vh, 17A);
668 label.ulft(btex PointRange etex, z22A);
669
670 %%% PointList (22, 13)
671 z22B=3/4[z22b1,z22t1];
```

```
672 VH(13bm,22B);
673 drawdashthreeO(22B, 13bm.vh, 13bm);
674 label.ulft(btex PointList etex, z22B);
675
676 %%% MeshLocation (22, 33)
677 z22C=1/4[z22br,z22tr];
678 z33B=2/3[z33b1,z33br];
679 VH(33B,22C);
680 drawnormalthreeO(22C, 33B.vh, 33B);
681 label.urt(btex (DER) Location etex, z22C);
682
683 %%% Meshloc (22, 33)
684 z22D=3/4[z22br,z22tr];
685 z33A=1/3[z33b1,z33br];
686 VH(33A,22D);
687 drawdashthreeO(22D, 33A.vh, 33A);
688 label.urt(btex Meshloc etex, z22D);
689
    The attributes of entity MeshConnectivity (z24).
690 %%% PointListSize (24, 23)
691 drawnormalO(24bm, 23tm);
692 label.ulft(btex PointListSize etex, z23tm);
693
694 %%% StructuredDonor (24, 34)
695 z24A=(x24br,y34ml);
696 draw z24A--z34ml dashes;
697 label.lrt(btex StructuredDonor etex, z24A);
698
699 %%% UnstructuredDonor (24, 35)
700 z24B=3/4[z24br,z24tr];
701 HvH(24B,35ml);
702 drawdashfourO(24B, 24B.hvh, 35ml.hvh, 35ml);
703 label.urt(btex UnstructuredDonor etex, z24B.hvh);
704
705 %%% Meshloc (24, 33)
706 z24C=4/6[z24b1,z24br];
707 z33C=(x33A,y33tm);
708 y2433C=1/3[y33C,y24C];
709 VyV(24C, 2433C, 33C);
710 drawdashfour0(24C, 24C.vyv, 33C.vyv, 33C);
711 label.lrt(btex Meshloc etex, z24C.vyv);
712
713 %%% Location (24, 33)
714 z24D=5/6[z24b1,z24br];
715 z33D=(x33B,y33tm);
716 y2433D=2/3[y33D,y24D];
717 VyV(24D, 2433D, 33D);
718 drawnormalfourO(24D, 24D.vyv, 33D.vyv, 33D);
719 label.lrt(btex (DER) Location etex, z24D.vyv);
```

```
720
721 %%% ConnectivityType (24, 36)
722 z24H=(x24D,y24tr);
723 z36H=z36bm;
724 y2436H=1/2[y35tr,y36br];
725 VyV(24H, 2436H, 36H);
726 drawnormalfourO(24H, 24H.vyv, 36H.vyv, 36H);
727 label.urt(btex (DER) ConnectivityType etex, z24H.vyv);
728
729 %%% Meshcon (24, 36)
730 z24G=(x24C,y24tr);
731 z36G=z36ml;
732 VH(24G,36G);
733 drawdashthreeO(24G, 24G.vh, 36G);
734 label.urt(btex Meshcon etex, z24G.vh);
735
736 %%% K (24, 27)
737 drawnormal0(24tm, 27bm);
738 label.ulft(btex K etex, z24tm);
739
740 %%% PointRange (24, 17)
741 z24E=1/4[z24t1,z24tr];
742 x17br-x17E = x17A-x17bl;
743 y 17E = y 17A;
744 VhV(24E,17E);
745 drawdashfour0(24E, 24E.vhv, 17E.vhv, 17E);
746 label.ulft(btex PointRange etex, z24E);
747
748 %%% PointList (24, 13)
749 z24F=1/4[z24bl,z24br];
750 z13F=1/2[z13t1,z13tr];
751 VhV(24F,13F);
752 drawnormalfourO(24F, 24F.vhv, 13F.vhv, 13F);
753 label.llft(btex PointList etex, z24F);
754
    The attributes of entity MeshConnectivity1_1 (z28)
755 %%% N (28, 27)
756 z27A=1/5[z27tl,z27tr];
757 z28A=(x27A,y28bl);
758 drawnormalO(28A,27A);
759 label.llft(btex N etex, z28A);
760
761 %%% trans (28, 27)
762 z27B=1/2[z27tl,z27tr];
763 z28B=(x27B,y28bl);
764 drawdash0(28B,27B);
765 label.lrt(btex \twolines{trans}{A[1:N]} etex, z28B);
766
767 %%% Transform (28, 27)
```

```
768 z27C=1/2[z27br,z27tr];
769 z28C=(1/2[x27C,x28br], y28br);
770 VH(28C,27C);
771 drawnormalthreeO(28C,28C.vh,27C);
772 label.lrt(btex (DER) Transform A[1:N] etex , z28C);
773
774 %%% DonorZone (28, 37)
775 z28D=1/2[z28br,z28tr];
776 z37D=z37tm;
777 VH(37D,28D);
778 drawdashthreeO(28D,37D.vh,37D);
779 label.urt(btex DonorZone etex, z28D);
780
781 %%% DonorPointRange (28, 17)
782 z28G=3/4[z28b1,z28t1];
783 z17G=(x17A,y17tl);
784 \text{ VH}(17G, 28G);
785 drawnormalthreeO(28G,17G.vh,17G);
786 label.ulft(btex DonorPointRange etex, z28G);
787
788 %%% PointRange (28, 17)
789 z28H=1/4[z28b1,z28t1];
790 z17H=(x17E,y17G);
791 VH(17H,28H);
792 drawnormalthreeO(28H,17H.vh,17H);
793 label.ulft(btex PointRange etex, z28H);
794
    The end of this diagram.
                                         % end fig 5
795 endfig;
796
```

3.3 Diagrams 6–11: Car model

The diagrams in this section are all taken from [SW94, Chapter 2]. A simple information/data model of a car, car model, and car manufacturer is presented in a variety of BLA languages, among which is, of course, EXPRESS-G. In words, the model is:

A car is made by a manufacturer. Each manufacturer has a unique name. A manufacturer constructs cars in several models and a car is of a particular model. A manufacturer gives a serial number to each car he produces and this is unique across all cars produced by that manufacturer. Each model also has a name, and this is unique across all models. A car has a year of production.

This model is a small portion of a conceptual model developed as a common specification for comparing modeling languages [ISO87].

The purpose of these diagrams is to illustrate how the express package supports a range of graphical languages.

3.3.1 Diagram 6: express-g

Having already seen several examples of EXPRESS-G diagrams, this one is presented with no further comments.

```
797
798 beginfig(6)
                           % example 6 EXPRESS-G
799 %%%drawgrid;
800
801 upshift := 3/2onelineh;
802 nextup := onelineh+upshift;
803
804 el := namespace(btex manufacturer etex)(nextra);
805 eh := onelineh;
806
807 hspace1 := 1/2el;
808
809 %%% y coords
810 lyc1 := 0;
811 lyc2 := lyc1+nextup;
812 lyc3 := lyc2+nextup;
813 lyc4 := lyc3+nextup;
814 lyc5 := lyc4+nextup;
815
816 %%% Zcol/row
817
818 %%% rows 4 and 5
819 z14=(0,lyc4);
820 drawENT(14, el, eh)(btex manufacturer etex);
821
822 z25=(x14br+hspace1, lyc5);
823 drawSTRING(25);
824
825 z34=(x25br+hspace1, lyc4);
826 drawENT(34, el, eh)(btex car\_model etex);
827
828 %%% row 3
829
830 z33bm=(x34bm, lyc3);
831 drawENT(33, el, eh)(btex car etex);
832
833 %%% row 2
834
835 z22=(x25, lyc2);
836 drawINTEGER(22);
837
838 %%% row 1
839
840 z21=(x22, lyc1);
841 drawSTRING(21);
842
```

```
843 %%% attributes
844
845 %%% manufacturer name (14, 25)
846 VH(14tm, 25ml);
847 drawnormalthreeO(14tm, 14tm.vh, 25ml);
848 label.urt(btex *name etex, z14tm.vh);
849
850 %%% model name (34, 25)
851 VH(34tm, 25mr);
852 drawnormalthreeO(34tm, 34tm.vh, 25mr);
853 label.ulft(btex *name etex, z34tm.vh);
854
855 %%% made_by (34, 14)
856 drawnormalO(34ml, 14mr);
857 label.ulft(btex made\_by etex, z34ml);
858
859 %%% model_type (33, 34)
860 drawnormalO(33tm, 34bm);
861 label.urt(btex model\_type etex, z33tm);
862
863 %%% DER made_by (33, 14)
864 VH(14bm, 33ml);
865 drawnormalthreeO(33ml, 14bm.vh, 14bm);
866 label.ulft(btex (DER) *made\_by etex, z33ml);
867
868 %%% year (33, 22)
869 z33A=1/3[z33bl, z33br];
870 VH(33A, 22mr);
871 drawnormalthreeO(33A, 33A.vh, 22mr);
872 label.ulft(btex year etex, z33A.vh);
873
874 %%% serial_no
875 z33B=2/3[z33b1, z33br];
876 VH(33B, 21mr);
877 drawnormalthreeO(33B, 33B.vh, 21mr);
878 label.ulft(btex *serial\_no etex, z33B.vh);
879
    The end of this diagram
880 endfig;
                          % end fig 6
881
```

3.3.2 Diagram 7: Shlaer-Mellor

The Shlaer-Mellor graphical language [SM88] is fairly simple, consisting basically of rectangles and arrowheaded lines. However, unlike EXPRESS-G each box usually has several lines of text. These are placed individually at the desired positions within each box.

Notice below that I have used **drawENT** with an empty string to draw the rectangular boxes; the routines for EXPRESS-G diagrams can, if suitable, be used

in any sort of diagram.

```
882
                           % example 7
883 beginfig(7)
884 %%%drawgrid;
885
886 el := namespace(btex MANUFACTURER etex)(niextra);
887 eh := onelineh;
888
889 hspace1 := namespace(btex made by maker of etex)(4ndextra);
890
891 hspace2 := 6onelineh;
                            % height of CAR
892 hspace3 := 5onelineh;
                            % space between rows
                            % height of MANUFACTURER
893 hspace4 := 3onelineh;
894
895 %%% y coords
896 lyc1 := 0;
897 lyc2 := lyc1+hspace2+hspace3;
898
899 %%% Zcol/row
900
901 %%% row 2
902
903 z12=(0, lyc2);
904 drawENT(12, el, hspace4)("");
905 x121=x12+sdtbs;
906 label.rt(btex MANUFACTURER etex, (x121, 2/3[y12bl, y12tl]));
907 label.rt(btex * manuf-name etex, (x121, 1/3[y12b1, y12t1]));
908
909 z22=(x12br+hspace1, lyc2);
910 drawENT(22, el, hspace4)("");
911 x221=x22+sdtbs;
912 label.rt(btex CAR-MODEL etex, (x221, 2/3[y12bl, y12tl]));
913 label.rt(btex * model-name etex, (x221, 1/3[y12b1, y12t1]));
914
915 %%% row 1
916
917 z21=(x22, lyc1);
918 drawENT(21, el, hspace2)("");
919 x211=x21+sdtbs;
                                    (x211, 5/6[y21bl, y21t1]));
920 label.rt(btex CAR etex,
921 label.rt(btex o year etex,
                                    (x211, 4/6[y21bl, y21tl]));
922 label.rt(btex o model (R) etex, (x211, 3/6[y21b1, y21t1]));
923 label.rt(btex * serial-no etex, (x211, 2/6[y21bl, y21tl]));
924 label.rt(btex * maker (R) etex, (x211, 1/6[y21bl, y21t1]));
925
926 %%%%% relationship lines
927
```

In this diagram the labels on the relationship lines tend to bump into the line ending symbol. METAPOST ignores initial and final spaces within btex...etex.

The paired braces below, each enclosing one space, supply non-discarded spaces. (These could also have been written as \space).

```
928 %%% MANUFACTURER / CAR-MODEL (12, 22)
929 drawnormalDCA(12mr, 22ml);
930 drawnormalCA(22ml, 12mr);
931 label.urt(btex { }{ }made by etex, z12mr);
932 label.llft(btex maker of{ }{ } etex, z22ml);
933
934 %%% MANUFACTURER / CAR (12, 21)
935 VH(12bm, 21ml);
936 drawnormalCA(12bm.vh, 12bm);
937 drawnormalDCA(12bm.vh, 21ml);
938 label.llft(btex { }{ }made by etex, z12bm);
939 label.ulft(btex maker of{ }{ } etex, z21ml);
940
941 %%% CAR-MODEL / CAR (22, 21)
942 drawnormalDCA(22bm, 21tm);
943 drawnormalCA(21tm, 22bm);
944 label.lrt(btex { }one of etex, z22bm);
945 label.ulft(btex type of{ } etex, z21tm);
946
    The end of this diagram
947 endfig;
                          % end fig 7
```

```
3.3.3 Diagram 8: IDEF1X
```

948

IDEF1X is a popular language in America for representing the structure of relational databases, and was developed under the auspices of the US Air Force [IDE85].

The general layout of the diagram mimics the previous one. All the boxes are divided horizontally into an upper and lower part. The **drawdoublerectangle** is used for the rectangular boxes, but the dividing line in the rounded box has to be added by hand, as it were.

```
949
                           % example 8 IDEF1X
950 beginfig(8)
951 %%%drawgrid;
952
953 el := namespace(btex MANUFACTURER etex)(niextra);
954 eh := onelineh;
955
956 hspace1 := namespace(btex made by maker of etex)(4ndextra);
957
958 hspace2 := 5onelineh;
                            % height of CAR
959 hspace3 := 5onelineh;
                            % space between rows
960 hspace4 := 3onelineh;
                            % height of MANUFACTURER
961
962 %%% y coords
```

```
963 lyc1 := 0;
964 lyc2 := lyc1+hspace2+hspace3;
965
966 %%% Zcol/row
967
968 %%% row 2
969
970 z12=(0, lyc2);
971 drawdoublerectangle(12, el, hspace4, 1/2);
972 label.urt(btex Manufacturer etex, z12tl);
973 x121=x12+sdtbs;
974 label.rt(btex manuf-name etex, (x121, 3/4[y12b1, y12t1]));
975
976 z22=(x12br+hspace1, lyc2);
977 drawdoublerectangle(22, el, hspace4, 1/2);
978 label.urt(btex Car-model etex, z22tl);
979 x221=x22+sdtbs;
980 label.rt(btex Model-name etex, (x221, 3/4[y12bl, y12tl]));
981 label.rt(btex Manuf-name (FK) etex, (x221, 1/4[y12b1, y12t1]));
982
983 %%% row 1
984
985 z21=(x22, lyc1);
986 drawroundedbox(21, el, hspace2, 2sdtbs)("");
987 label.urt(btex Car etex, z21tl);
988 draw z21ml--z21mr;
989 x211=x21+sdtbs;
                                         (x211, 5/6[y21bl, y21tl]));
990 label.rt(btex Serial-no etex,
991 label.rt(btex Manuf-name (FK) etex, (x211, 4/6[y21bl, y21t1]));
992 label.rt(btex Model-name (FK) etex, (x211, 2/6[y21bl, y21tl]));
993 label.rt(btex Year etex,
                                         (x211, 1/6[y21bl, y21tl]));
994
995 %%%%% relationship lines
996
997 %%% MANUFACTURER / CAR-MODEL (12, 22)
998 drawnormalD(12mr, 22ml);
999 label.top(btex of etex, 1/2[z12mr, z22ml]);
1000
1001 %%% MANUFACTURER / CAR (12, 21)
1002 VH(12bm, 21ml);
1003 drawnormalthreeD(12bm, 12bm.vh, 21ml);
1004 label.top(btex makes etex, 1/2[z12bm.vh, z21ml]);
1005 label.ulft(btex P{ } etex, z21ml);
1006
1007 %%% CAR-MODEL / CAR (22, 21)
1008 drawnormalD(22bm, 21tm);
1009 label.lft(btex of etex, 1/2[z21tm,z22bm]);
1010 label.urt(btex { }P etex, z21tm);
1011
```

The end of this diagram 1012 endfig; % end fig 8 1013

3.3.4 Diagram 9: OMT

OMT [RBP+91] was the precursor to the currently popular UML structure modeling language. If you are curious as to why I believe that lexical languages are in general better than BLA languages, the chapter on developing a compiler for OMT highlights, probably unintentionally, some of the disadvantages of BLAs.

The general look of the OMT diagram is almost identical to the IDEF1X diagram.

```
1014
                            % example 9 OMT
1015 beginfig(9)
1016 %%%drawgrid;
1017
1018 el := namespace(btex {\bf MANUFACTURER} etex)(niextra);
1019 eh := onelineh;
1020
1021 hspace1 := namespace(btex made by maker of etex)(4ndextra);
1022
1023 hspace2 := 4onelineh;
                             % height of CAR
1024 hspace3 := 4onelineh;
                             % space between rows
1025 hspace4 := 3onelineh;
                             % height of MANUFACTURER
1026
1027 %%% y coords
1028 lyc1 := 0;
1029 lyc2 := lyc1+hspace2+hspace3;
1030
1031 %%% Zcol/row
1032
1033 %%% row 2
1034
1035 z12=(0, lyc2);
1036 drawdoublerectangle(12, el, hspace4, 1/2);
1037 x121=x12+sdtbs;
1038 label.rt(btex {\bf Manufacturer} etex, (x121, 3/4[y12b1, y12t1]));
1039 label.rt(btex name : String etex,
                                            (x121, 1/4[y12b1, y12t1]));
1040
1041 z22=(x12br+hspace1, lyc2);
1042 drawdoublerectangle(22, el, hspace4, 1/2);
1043 x221=x22+sdtbs;
1044 label.rt(btex {\bf Car Model} etex, (x221, 3/4[y22b1, y22t1]));
1045 label.rt(btex name: String etex,
                                        (x221, 1/4[y22bl, y22tl]));
1046
1047 %%% row 1
1048
1049 z21=(x22, lyc1);
1050 drawdoublerectangle(21, el, hspace2, 1/3);
```

```
1051 x211=x21+sdtbs;
1052 label.rt(btex {\bf Car} etex,
                                             (x211, 5/6[y21bl, y21t1]));
1053 label.rt(btex serial-no : String etex, (x211, 4/9[y21bl, y21tl]));
1054 label.rt(btex year : Integer etex,
                                            (x211, 2/9[y21bl, y21t1]));
1055
1056 %%%%% relationship lines
1057
1058 %%% MANUFACTURER / CAR-MODEL (12, 22)
1059 drawnormalD(12mr, 22ml);
1060 label.top(btex {\it makes} etex, 1/2[z12mr, z22ml]);
1061 label.ulft(btex 1+{ } etex, z22ml);
1062
1063 %%% MANUFACTURER / CAR (12, 21)
1064 VH(12bm, 21ml);
1065 drawnormalthreeD(12bm, 12bm.vh, 21ml);
1066 label.top(btex {\it makes} etex, 1/2[z12bm.vh, z21ml]);
1067 label.ulft(btex 1+{ } etex, z21ml);
1068
1069 %%% CAR-MODEL / CAR (22, 21)
1070 drawnormalD(22bm, 21tm);
1071 label.lft(btex {\it produced as} etex, 1/2[z21tm,z22bm]);
1072
     The end of this diagram
```

```
1073 endfig; % end fig 9
1074
```

3.3.5 Diagram 10: E-R

There are several flavours of Entity-Relationship modeling icons. I have used a style from [EN89].

This example introduces some new icons and METAPOST modeling.

```
1075
1076 beginfig(10)
                             % example 10 E-R
1077 %%%drawgrid;
1078
1079 numeric diam; diam := 2onelineh;
1080
1081 upshift := diam;
                                  % circle diameter
1082 nextup := diam+upshift;
1083
1084 el := namespace(btex MANUFACTURER etex)(niextra);
1085 eh := onelineh;
1086
1087 numeric dl, dh;
                                  % length and height of diamond boxes
1088 dl := 1/2el; dh := 2eh;
1089
1090 numeric marg; marg := 4u;
                                 % double the seperation between parallel lines
1091
```

```
1092 hspace1 := 1/2el;
1093
1094 %%% y coords
1095 lyc1 := 0;
1096 lyc2 := lyc1+nextup;
1097 lyc3 := lyc2+nextup;
1098 lyc4 := lyc3+nextup;
1099 lyc5 := lyc4+nextup;
1100
```

The diagram has 5 rows. Start with the 4th as it is the widest row.

```
1101 %%% Zcol/row
1102 %%% row 4
1103
1104 z14=(0,lyc4);
1105 drawENT(14, el, eh)(btex manufacturer etex);
1106
1107 z34ml=(x14br+hspace1, y14mr);
1108 drawdiamondbox(34, dl, dh)(btex has etex);
1109
1110 z54ml=(x34mr+hspace1, y34mr);
1111 drawENT(54, el, eh)(btex car\_model etex);
1112
1113 %%% row 5
1114
1115 z15bm=(x14bm, lyc5);
1116 drawcirclebox(15, diam)(btex Name etex);
1117
1118 z55bm=(x54bm, lyc5);
1119 drawcirclebox(55, diam)(btex Name etex);
1120
```

The third row consists of an ordinary diamond box at the right and a doubly enclosed diamond on the left. The drawtwodiamondbox routine is used for the latter. Note that values of marg are used to make the oustside of the double diamond larger than the single diamonds.

```
1121 %%% row 3
1122
1123 z53bm=(x54bm, lyc3);
1124 drawdiamondbox(53, dl, dh)(btex of etex);
1125
1126 z13c=(x14bm, y53c);
1127 drawtwodiamondbox(13, dl+marg, dh+marg, 2/5marg)(btex Makes etex);
1128
```

The second row just consists of a doubly enclosed rectangular box. There is no explicit routine for drawing this, so two **drawENT** routines are used for drawing smaller and larger superimposed boxes (one with an empty text argument).

1129 **%%% row 2**

1130

```
1131 z32bm=(x34bm, lyc2);
1132 drawENT(32, el, eh)(btex car etex);
1133 z31c=z32c;
1134 drawENT(31, el+marg, eh+marg)("");
1135
1136 %%% row 1
1137
1138 z21bm=(x32bl, lyc1);
1139 drawcirclebox(21, 5/4diam)(btex \twolines{Serial}{No} etex);
1140
1141 z41bm=(x32br, lyc1);
1142 drawcirclebox(41, diam)(btex Year etex);
1143
1144 %%%%%% lines
1145
1146 %%% Name / manufacturer (15, 14)
1147 drawnormal(15bm, 14tm);
1148
1149 %%% Name / model (55, 54)
1150 drawnormal(55bm, 54tm);
1151
1152 %%% Mnf / has (14, 34)
1153 drawnormal(14mr, 34ml);
1154 label.top(btex 1 etex, 1/2[z14mr,z34ml]);
1155
1156 %%% has/ model (34, 54)
1157 drawnormal(34mr, 54ml);
1158 label.top(btex N etex, 1/2[z34mr,z54ml]);
1159
1160 %%% Mnf / makes (14, 13)
1161 drawnormal(14bm, 13tm);
1162 label.rt(btex 1 etex, 1/2[z14bm,z13tm]);
1163
1164 %%% model / of (54, 53)
1165 drawnormal(54bm, 53tm);
1166 label.rt(btex 1 etex, 1/2[z54bm,z53tm]);
1167
```

In the diagrams so far, except trivially in the first, the lines have all been vertical and/or horizontal. Now we meet lines in arbitrary directions, and parallel to boot.

A pair of parallel lines joins the bottom of the doubly enclosed diamond to the left side of the doubly enclosed rectangle. Drawing one of these is simple — just draw a straight line between the bottom middle of the diamond to the middle left of the rectangle.

1168 %%% makes / car (13, 31) 1169 drawnormal(13bm, 31ml);

The point z31A is 1/2marg above the point z31ml. This is one end of the parallel line.

1170 z31A=(x31ml, y31ml+1/2marg);

The METAPOST equation z1-z2 = frac*(z3-z4) means that (a) the line z1-z2 is parallel to the line z3-z4, and (b) the length of the line z1-z2 is frac of the length of z3-z4. We next specify that the line z13U-z31A is parallel to, and the same length as, the line z13bm-z31ml;

1171 z13U-z31A = z13bm-z31ml;

The final part of the calculation is to determine the point z13A so that the line z31A--z13A is parallel to our first line and starts and ends at the outer perimeters of the boxes (and we already know what z31A is).

Remember mediation? A point, z1234 is on the line z1--z2 if

z1234=frac1[z1,z2 holds.

whatever

Similarly a point z1234 is on the line z3--z4 if

z1234=frac2[z3,z4] holds.

These two equations are linear and provided enough variables are known it is easy, even if tedious and error prone, to solve for all the unknowns using simple mathematics. Fortunately METAPOST has an idiom for this kind of calculation using the built-in variable called whatever. Each time whatever is used it is internally given a new name which saves a lot of name inventions on the part of the user. The following line of code results in METAPOST calculating the intersection point, which I call z13A, between the two lines z13U,z31A, which is our parallel lines whose points are known, and the line z13bm--z13mr which is the lower right line of the outer diamond whose points are known.

1172 z13A = whatever[z13U,z31A] = whatever[z13bm,z13mr];

Finally, draw the second parallel line in the desired position and length.

```
1173 draw z13A--z31A;
1174 label.urt(btex N etex, 1/2[z13A,z31A]);
1175
```

Another pair of parallel lines connects the doubly bounded rectangle and the singly bounded diamond, and we use a similar calculation as above.

```
1176 %%% of / car (53, 31)
1177 drawnormal(53bm, 31mr);
1178 z31B=(x31mr, y31mr+1/2marg);
1179 z53U - z31B = z53bm - z31mr;
1180 z53B = whatever[z53U,z31B] = whatever[z53bm,z53ml];
1181 draw z53B--z31B;
1182 label.ulft(btex N etex, 1/2[z53B,z31B]);
1183
```

That's the end of the tricky bits for this diagram.

```
1184 %%% car / serial no (31, 21)
1185 z31C=(x21tm, y31bl);
1186 drawnormal(31C, 21tm);
1187
1188 %%% car / year (31, 41)
1189 z31D=(x41tm, y31br);
1190 drawnormal(31D, 41tm);
1191
```

The end of this diagram

1192 endfig; % end fig 10

3.3.6 Diagram 11: NIAM

Like IDEF1X, NIAM [NH89] is a language for relational database structures. In Europe it tends to be preferred to IDEF1X.

Although the layout and icons of the NIAM illustration differ from the prior diagrams, no new techniques are used. The most obvious of the differences are the use of oval and circular boxes in place of the rectangular boxes, and the use of pairs of rectangular boxes in the middle of the relationship lines.

```
1194
1195 beginfig(11)
                             % example 11 NIAM
1196 %%%drawgrid;
1197
1198 numeric diam; diam := 2onelineh;
                                          % circle diameter
1199
1200 upshift := diam;
1201 nextup := diam+upshift;
1202
1203 numeric hel, vel;
                              % length of horizontal and vertical box pairs
1204 eh := onelineh;
1205 hel = namespace(btex made by etex)(nextra);
1206 vel = namespace(btex of prod etex)(nextra);
1207
                              % horizontal & vertical diameters of oval boxes
1208 numeric del, deh;
1209 del := namespace(btex MANUFACTURER etex)(niextra);
1210 deh := 2eh;
1211
1212 numeric marg; marg := 2u;
                                      % margin for doubly enclosed oval boxes
1213 numeric lmarg; lmarg := marg;
                                      % gap for short lines
1214
1215 numeric dashel, dasheh;
                                      % diameters of dashed ellipses
1216 dashel := del+2marg;
1217 dasheh := deh+2marg;
1218
1219 hspace1 := diam;
1220
1221 %%% y coords
1222 lyc1 := 0;
1223 lyc2 := lyc1+nextup;
1224 lyc3 := lyc2+nextup;
1225
1226 %%% Zcol/row
1227
1228 %%% row 3
1229
1230 x13ml=0; y13bm=lyc3;
```

```
1231 drawovalbox(13, del, deh)(btex manufacturer etex);
1232 z14=z13c;
1233 drawdashellipse(14, dashel, dasheh);
1234
1235 z23ml=(x14mr+hspace1, y14mr);
1236 drawENT(23, hel, eh)(btex of etex);
1237
1238 z33bl=z23br;
1239 drawENT(33, hel, eh)(btex made by etex);
1240
1241 z44ml=(x33mr+hspace1, y33mr);
1242 drawdashellipse(44, dashel, dasheh);
1243 z43=z44c;
1244 drawovalbox(43, del, deh)(btex car model etex);
1245
1246 z53ml=(x44mr+hspace1, y44mr);
1247 drawdashcircle(53, diam);
1248 label(btex year etex, z53c);
1249
1250 %%% row 2
1251
1252 z22c=(x23bm,lyc2);
1253 drawcirclebox(22, namespace(btex U etex)(2nextra))(btex U etex);
1254
1255 x42tm=x43bm; y42ml=lyc2;
1256 drawdoublerectangle(42, vel, 2eh, 1/2);
1257 label(btex of etex, z42ct);
1258 label(btex is of etex, z42cb);
1259
1260 %%z52c=(x53x,lyc2);
1261 x52tm=x53bm; y52ml=lyc2;
1262 drawdoublerectangle(52, vel, 2eh, 1/2);
1263 label(btex of prod etex, z52ct);
1264 label(btex prod in etex, z52cb);
1265
1266 %%% row 1
1267
1268 z11ml=(x13ml,lyc1);
1269 drawdashellipse(11, del, deh);
1270 label(btex serial no etex, z11c);
1271
1272 z21ml=(x23ml,lyc1);
1273 drawENT(21, hel, eh)(btex of etex);
1274
1275 z31bl=z21br;
1276 drawENT(31, hel, eh)(btex has etex);
1277
1278 z41c=(x43c,lyc1);
1279 drawcirclebox(41, diam)(btex car etex);
1280
```

```
1281 %%%%%% lines
1282
1283 %%% manufacturer / of (14, 23)
1284 drawnormal(14mr, 23ml);
1285
1286 %%% made by / car model (33, 44)
1287 drawnormal(33mr, 44ml);
1288 draw (x33tl, y33tl+lmarg)--(x33tr, y33tl+lmarg);
1289 draw (x33mr, y33mr+lmarg)--(x33mr+eh, y33mr+lmarg);
1290
1291 %%% serial no / of (11, 21)
1292 drawnormal(11mr, 21ml);
1293
1294 %%% of / of (23, 22, 21)
1295 drawnormal(23bm, 22tm); drawnormal(22bm, 21tm);
1296 label.rt(btex id etex, z22mr);
1297
1298 %%% has / car (31, 41)
1299 drawnormal(31mr, 41ml);
1300 draw (x31tl, y31tl+lmarg)--(x31tr, y31tl+lmarg);
1301 draw (x31mr, y31mr+lmarg)--(x33mr+eh, y31mr+lmarg);
1302
1303 %%% model / of (44, 42)
1304 drawnormal(43bm, 42tm);
1305
1306 %%% is of / car (42, 41)
1307 drawnormal(42bm, 41tm);
1308 draw (x42tfr+lmarg, y42tfr)--(x42tfr+lmarg, y42br);
1309 draw (x42bm+lmarg, y42bm)--(x42bm+lmarg, y42bm-eh);
1310
1311 %%% year / of prod (53, 52)
1312 drawnormal(53bm, 52tm);
1313
1314 %%% prod in / car (52, 41)
1315 VH(52bm,41mr);
1316 drawnormalthree(52bm, 52bm.vh, 41mr);
1317 draw (x52tfr+lmarg, y52tfr)--(x52tfr+lmarg, y52br);
1318 draw (x52bm+lmarg, y52bm)--(x52bm+lmarg, y52bm-eh);
1319
     The end of this diagram
```

1320 endfig; % end fig 11 1321

Note that the centers of the boxes on the bottom row are all at y=0, and hence the bottom halves have negative y values. This does not matter as METAPOST determines the actual space required for a diagram and does not base it on the origin of the coordinate system.

The end of the METAPOST examples file.

1322 end

The end of the example.

1324 $\langle / eg \rangle$

3.4 Comments

Everyone is likely to have a different style of diagramming. You have seen how I do it but that might not be what suits you. Nevertheless, the following comments may be of assistance.

In my own work I tend to draw a lot of diagrams, using different .mp files, that have similar sets of 'commonly used variables & values'. Rather than repeating the list in each file, I create a METAPOST file, called say mycommons.mp, that includes these and then input mycommons after the input expresses near the start of each diagram file.

I find it a help to sketch out the diagrams using pencil and paper before coding them up. I have to admit, though, that the final results often look very different from my initial ideas of the layout.

You might have noticed that I start the coding at different places in the diagrams. Basically I try and plan on getting the boxes aligned roughly in rows and/or columns. I then typically start with the longest column, or more often with the widest row, and then fill in the rest from there.

3.5 Running the MetaPost example file

The expeg.mp file has to be processed by METAPOST to obtain Encapsulated PostScript (eps) files for viewing or printing. Depending on your installation the command to run METAPOST may be either:

mpost expeg or

${\tt mp}$ expeg

or perhaps something similar. This will produce eleven files, expeg.1 to expeg.11, each of which is an eps file for the corresponding beginfig(N) in expeg.mp. METAPOST defaults to using TEX for processing labels which outputs them using Computer Modern fonts. Unfortunately PostScript applications do not understand these so it is necessary to get the fonts inserted. This is most simply done by $I^{AT}EX$ and document that includes the METAPOST output. A suitable file is provided below and is configured to be processed by either $I^{AT}EX$ or $pdfI^{AT}EX$. Run $I^{AT}EX$ and then dvips (or what you normally use to generate printable output) and you will get all eleven diagrams.

1325 $\langle *egt \rangle$

```
1326 %%% expeg.tex display expressg.dtx MetaPost examples
1327
1328 \documentclass[11pt]{article}
1329 \newif\ifpdf
1330 \ifx\pdfoutput\undefined
1331 \pdffalse
```

1323

```
1332 \else
1333 \pdftrue
1334 \fi
1335
1336 \ifpdf
1337
     \pdfoutput=1
1338
      \usepackage[pdftex,final]{graphicx}
     \DeclareGraphicsRule{*}{mps}{*}{}
1339
1340 \else
1341 \usepackage[final]{graphicx}
1342 \fi
1343
1344 %%%% page sizes for ISO document on A4 paper
1345 \setlength{\headheight}{11pt}
1346 \setlength{\headsep}{10mm}
1347 \setlength{\topskip}{11pt}
1348 \setlength{\footskip}{11mm}
1349 \setlength{\textwidth}{160mm}
1350 \setlength{\textheight}{221.5mm}
1351 \setlength{\columnsep}{10mm}
1352 \setlength{\topmargin}{0mm}
1353 \setlength{\oddsidemargin}{0mm}
1354 \setlength{\evensidemargin}{0mm}
1355 \setlength{\marginparwidth}{0pt}
1356 \setlength{\marginparsep}{0pt}
1357 \setlength{\marginparpush}{0pt}
1358 \setlength{\footnotesep}{12pt}
      %%%% for US letterpaper need to change some margins
1359
1360 \setlength{\topmargin}{-9.4mm}
1361 \setlength{\oddsidemargin}{1.55mm}
1362 \setlength{\evensidemargin}{1.55mm}
1363
1364 \begin{document}
1365
1366 \begin{figure}
1367 \centering
1368 \includegraphics{expeg.1}
1369 \caption{Some boxes and line styles}
1370 \end{figure}
1371
1372 \begin{figure}
1373 \centering
1374 \includegraphics{expeg.2}
1375 \caption{Example schema level diagram}
1376 \end{figure}
1377
1378 \begin{figure}
1379 \centering
1380 \includegraphics{expeg.3}
1381 \caption{Example diagram of a tree structure}
```

```
1382 \end{figure}
1383
1384 \begin{figure}
1385 \centering
1386 \includegraphics{expeg.4}
1387 \caption{Supertypes and subtypes}
1388 \end{figure}
1389
1390 \begin{figure}
1391 \centering
1392 \includegraphics{expeg.5}
1393 \caption{A portion of a large model}
1394 \end{figure}
1395
1396 \begin{figure}
1397 \centering
1398 \includegraphics{expeg.6}
1399 \caption{Car model using EXPRESS-G}
1400 \end{figure}
1401
1402 \begin{figure}
1403 \ centering
1405 \ Car model using Shlaer-Mellor\}
1406 \end{figure}
1407
1408 \begin{figure}
1409 \centering
1410 \includegraphics{expeg.8}
1411 \caption{Car model using IDEF1X}
1412 \end{figure}
1413
1414 \begin{figure}
1415 \centering
1416 \includegraphics{expeg.9}
1417 \caption{Car model using OMT}
1418 \end{figure}
1419
1420 \begin{figure}
1421 \centering
1422 \includegraphics{expeg.10}
1423 \caption{Car model using E-R}
1424 \end{figure}
1425
1426 \begin{figure}
1427 \setminus centering
1428 \includegraphics{expeg.11}
1429 \caption{Car model using NIAM}
1430 \end{figure}
1431
```

```
1432 \end{document}
1433
```

The end of the IAT_EX file.

1434 $\langle / \mathsf{egt} \rangle$

3.5.1 Using pdfIATEX

pdflAT_EX generates a .pdf file directly from LAT_EX source instead of a .dvi file. Unfortunately pdfLAT_EX does not understand Encapsulated PostScript in general, but it *does* understand the restricted form of Encapsulated PostScript generated by METAPOST. By default pdfLAT_EX will recognise files with an .mps extension as coming from METAPOST, but METAPOST generates files with a numeric extension. If you like to use both METAPOST and pdfLAT_EX, then it can get tedious changing all the numeric extensions to .mps. The following two scripts can help with this. The first is a shell script⁴ which calls a Perl script.

The shell script is called n2mps.sh. To get suitable versions of the example METAPOST output files for use with $pdfIAT_EX$, just do:

n2mps.sh expeg

after having run METAPOST on expeg.mp.

```
1435 \langle * \mathsf{shell} \rangle
1436 #! /bin/sh
1437
1439 # Shell script n2mps.sh
1440 # Call as: n2mps.sh basename
1441 # List each file basename.* in the directory and run the Perl script
1442 # to copy each basename.N to basenameN.mps, where N is an integer
1443 #
1444 # Copyright 2000, Mauro S. Costa and Peter R. Wilson
1446
1447 basename=${1:?"A file basename is required."}
1448 extname=N.mps
1449 echo Files $basename.N, where N is a number, will be copied to $basename$extname
1450 for file in 'ls $basename.*'
1451 do
1452 n2mpsprl.prl $file
1453 done
1454
1456
1457 \langle / \text{shell} \rangle
 The Perl script, which is called by the n2mps script:
1458 (*perl)
1459
```

⁴You may well have to modify this to run on your operating system.

```
1460 #!/usr/local/bin/perl -w
1461
1463 # Perl script: n2mpsprl.prl
1464 # Call as: n2mpsprl.prl filename
1465 # If filename is of the form basename.N, where N is an integer,
1466 # copies file basename.N to file basenameN.mps
1467 #
1468 # Copyright 2000, Mauro S. Costa and Peter R. Wilson
1470
1471 # Test for correct number of input parameters
1472 die "Invalid command line arguments.\nTry $0 <src> \n" if($#ARGV > 1);
1473 die "Invalid command line arguments.\nTry $0 <src> \n" if($#ARGV < 0);
1474
1475 # Assign input file name to variable
1476 $input_file = $ARGV[0];
1477
1478 ## test if ends with a number, exit if not
1479 if ($input_file = ~ /\w\.\d/) { ; } else { exit; }
1480
1481 # Remove the "dot" from the string variable
1482 # holding the input file name
1483 sinput_file = s/.// ;
1484
1485 # Create a list variable composed of the string variable holding
1486 # the concatenated input file name and the extension ".mps"
1487 @name_list = ($input_file,'.mps');
1488
1489 # Join the string variables in the name_list variable into
1490 # a single string variable
1491 $output_file = join("",@name_list) ;
1492
1493 # create a list variable composed to the parameters needed
1494 # for the system copy command excution
1495 @exec_list = ('cp', $ARGV[0], $output_file);
1496
1497 # Execute the system copy ("cp") command
1498 system(@exec_list) ;
1499
1501
1502 \langle / perl \rangle
```

As an alternative when using pdfLATEX and the graphicx package, putting \DeclareGraphicsRule{*}{mps}{*}{} will cause included graphics files with unknown extensions (e.g., as generated by METAPOST) to be treated as .mps files.

```
\documentclass...
\usepackage{ifpdf}% you should have this
```

```
\ifpdf
  \pdfoutput=1
  \usepackage[pdftex,final]{graphicx}
  \DeclareGraphicsRule{*}{mps}{*}{}
  \else
    \usepackage[final]{graphicx}
  \fi
  ...
  \includegraphics{mpfig.23}
  ...
```

3.6 Using LATEX instead of TEX

As already noted, METAPOST defaults to using T_EX for typesetting labels. If you would prefer it to use IAT_FX instead, then two things have to be done.

1. METAPOST looks at the value of an environment variable called TEX to see what typesetting system it should use. If the variable is not set, then it uses T_EX . To get it to use IAT_EX the environment variable has to be set to latex. For example, on the system I use this is done by:

TEX=latex export TEX

2. LATEX \documentclass and \begin{document} commands must be put into the verbatimtex...etex group at the start of the diagramming file. For the example, this might look like

```
verbatimtex
  \documentclass{article}
  % \usepackage{...} % for any packages
  \def\twolines#1#2{\vbox{\hbox{#1} \hbox{#2}}}
  \begin{document}
etex
```

NOTE: Both the above are required for $\mathbb{IAT}_{E}X$. On the other hand, if **TEX** is either undefined or has a value other than **tex**, then there must be no $\mathbb{IAT}_{E}X$ code anywhere in the .mp file.

If you need to generate individual self-contained **ps** files for inclusion in non-LATEX documents where the font used for labels is a member of the Computer Modern family, then you need to use LaTeX and dvips on ***.tex** files that consist of only a single diagram and no other text at all. At least on my setup I have found it best to run dvips as normal but with output to a file instead of a printer. Then use whatever tools are available to you to convert the .ps file to an .eps one (or at least set the correct bounding box values). The primary source on generating and using Encapsulated PostScript in the LATEX world is [Rec97], and [GRM97] also has useful information.

3.7 Using PostScript fonts

METAPOST can be made to use PostScript fonts and, providing the labels don't require any special (T_EX) processing, then use of LATEX may be avoided altogether. In order to do this, certain requirements have to be put on the content on the .mp file.

- The diagram file must begin with the line prologues:=2; which will add a prologue for PostScript font to the output file.
- The defaultfont must be changed. What it has to be changed to depends on the font you want to use. For example, to use the Times Roman font, put:

defaultfont:="ptmr8r";
just after the prologues line.

For this incantation to work, the file ptmr8r.tfm must exist in a location where METAPOST looks for .tfm files (on the system I use it is in directory ../texmf/fonts/tfm/adobe/times). Also, the file psfonts.map must include a line like

ptmr8r Times-Roman ...

Again, on my system psfonts.map is in directory .../texmf/dvips/misc.

I don't know whether or not this works if you don't have a LATEX distribution — it probably doesn't. But it's unlikely that you don't have one if you intend to use METAPOST.

As a final note on changing fonts, the METAPOST defaultfont and defaultscale values only apply to quoted text (e.g., "quoted text") and not to 'btexed' text (e.g., btex btexed text etex). Conversely, T_EX or LAT_EX commands only apply to btexed text and not to quoted text. For more details on this, see [Wil99].

4 The MetaPost code

We start by announcing what it is for.

```
1503 (*up)
1504 %%% EXPRESSG.MP MetaPost macros for EXPRESS-G and other BLA diagrams
1505 %%% version 1.5, 31 July 2003
1506 %%% version 1.6, 29 February 2004
1507 %%% version 1.61, 17 March 2004
1508
1509 show "expressg.mp version 1.61, 2004/03/17";
1510
```

4.1 Variables

u The unit of length.

```
1511 newinternal u; numeric u;
                 1512 u := 1mm;
                 1513
            maxx The maximum x and y coordinates for a diagram.
            maxy 1514 newinternal maxx, maxy;
                 1515 numeric maxx, maxy;
                                             % smidgeon under 160mm for ISO
                 1516 maxx := 159.5u;
                 1517 maxy := 210u;
                                             % 210mm for ISO
                 1518
  defaultdotdiam The default and actual diameter of circle and dot line end styles.
         dotdiam 1519 newinternal defaultdotdiam, dotdiam;
                 1520 numeric defaultdotdiam, dotdiam;
                 1521 defaultdotdiam := 2u;
                 1522 dotdiam := defaultdotdiam;
                 1523
   normalpensize The diameters of the pens for drawing normal, thick and thin lines. (The size of
   normalpensize the default METAPOST pen is 0.5bp).
   normalpensize 1524 newinternal normalpensize, thickpensize, thinpensize;
                 1525 numeric normalpensize, thickpensize, thinpensize;
                 1526 normalpensize := 0.5bp;
                 1527 thickpensize := 1.5bp;
                 1528 thinpensize := 0.25bp;
       dotsscale dotsscale is the amount the ...pensize has to be increased so that a dotted
                  line looks as thick as a continuous or dashed line.
                 1529 newinternal dotsscale; numeric dotsscale;
                 1530 dotsscale := 2;
                 1531
       normalpen The pens for drawing lines, plus dotpen for drawing a dot line end style.
        thickpen 1532 newinternal normalpen, thickpen, thinpen, dotspen, dotpen;
         thinpen 1533 pen normalpen, thickpen, thinpen, dotspen, dotpen;
         dotspen 1534 normalpen := pencircle scaled normalpensize; % i.e. defaultpen;
          dotpen 1535 thickpen := pencircle scaled thickpensize;
                 1536 thinpen := pencircle scaled thinpensize;
                 1537 dotspen := pencircle scaled (dotsscale*normalpensize);
                 1538 dotpen
                               := pencircle scaled dotdiam;
                 1539
defaultsmoothrad The default and initial values for the radius of the arc used for joining two lines.
       smoothrad 1540 newinternal defaultsmoothrad;
                 1541 numeric smoothrad, defaultsmoothrad;
                 1542 smoothrad := defaultsmoothrad := 2u;
                 1543
```

```
defaultdrumlid The default and initial values for the ratio of the minor to major diameter of
       drumlid ellipses for the top and bottom of drums.
               1544 newinternal defaultdrumlid;
               1545 numeric drumlid, defaultdrumlid;
               1546 drumlid := defaultdrumlid := 0.2;
               1547
    defaultgal The default values for the length and base width width of arrowheads and fanins.
    defaultgab _{1548}\mbox{ \sc width} for arrowheads and fanin
    defaultgfl<sub>1549</sub> newinternal defaultgal, defaultgab, defaultgfl, defaultgfb;
    defaultgfb 1550 numeric defaultgal, defaultgab, defaultgfl, defaultgfb;
               1551 defaultgal := defaultgfl := defaultdotdiam;
               1552 defaultgab := defaultgfb := defaultgal;
               1553
           gal Values of length and base width of arrowheads and fanins.
           gab 1554 %/% length and base width of arrowheads and fanins
           gfl 1555 newinternal gal, gab, gfl, gfb;
           gfb 1556 numeric gal, gab, gfl, gfb;
               1557 gal := defaultgal;
               1558 gab := defaultgab;
               1559 gfl := defaultgfl;
               1560 gfb := defaultgfb;
               1561
      onelineh The minimum height of a box that encloses a single line of text.
```

```
1562 newinternal onelineh;
1563 numeric onelineh;
1564 onelineh := 5u;
1565
```

```
sdtbl The length, height and inset for a simple data type box. The length is sufficientsdtbh for the name 'BOOLEAN' and the height for one line of text.
```

```
sdtbs 1566 %%% length, height & inset for simple data type boxes
1567 newinternal sdtbl, sdtbh, sdtbs;
1568 numeric sdtbl, sdtbh, sdtbs;
1569 sdtbl := 22u;
1570 sdtbh := onelineh;
1571 sdtbs := 2u;
1572
```

sdtbel The length, height and inset for a simple EXPRESSION data type box. The sdtbeh length is sufficient for the name 'EXPRESSION' and the height for one line of sdtbes text.

```
1573 %%% length, height & inset for EXPRESSION data type boxes
1574 newinternal sdtbel, sdtbeh, sdtbes;
1575 numeric sdtbel, sdtbeh, sdtbes;
1576 sdtbel := 28u;
```

```
1577 sdtbeh := sdtbh; sdtbes := sdtbs;
       1578
sdtbgel The length, height and inset for a simple GENERICENT data type box. The
sdtbgeh length is sufficient for the name 'GENERIC_ENTITY' and the height for one line
sdtbges of text.
       1579 %%% length, height & inset for GENERICENT data type boxes
       1580 newinternal sdtbgel, sdtbgeh, sdtbges;
       1581 numeric sdtbgel, sdtbgeh, sdtbges;
       1582 sdtbgel := 38u;
       1583 sdtbgeh := sdtbh; sdtbges := sdtbs;
       1584
 pconl The average length of a numeric page connector (e.g., for 9,9 (9,9)).
       1585 %%% average length of numeric page connector
       1586 newinternal pconl;
       1587 numeric pconl; % length of page connector (for e.g., 9,9 (9,9) )
       1588 pconl:=15u;
  pconh Height of average page connector (one line of text).
       1589 %%% height of page connectors (one line)
       1590 newinternal pconh;
       1591 numeric pconh;
       1592 pconh := onelineh;
       1593
   enth The height of a one line ENT(ITY) box (enth) and the height of a one line
  typeh ENUM(ERATION), SELECT or TYPE box.
       1594 %%% heights of entity, enum, select, type boxes (One text line)
       1595 newinternal enth, typeh;
       1596 numeric enth, typeh;
       1597 enth := onelineh;
       1598 typeh := onelineh;
       1599
   ish The height of a normal one line interschema box (ish) and an interschema box
   isrh with one line each for the name and rename (isrh).
       1600 %%% height of interschema boxes (one text line, no rename), and (name + rename)
       1601 newinternal ish, isrh;
       1602 numeric ish, isrh;
       1603 ish := 2onelineh;
       1604 isrh := 3onelineh;
       1605
schemah Height of a one line SCHEMA box.
       1606 %%% height of schema boxes (one text line)
       1607 newinternal schemah;
       1608 numeric schemah;
       1609 schemah := 2onelineh;
       1610
```

```
eventh The height (eventh) of a one line (G/L)EVENT box and the slope of the sides of
eventslope a (G/L)EVENT box.
          1611 %%% height and slope of event boxes
          1612 newinternal eventh, eventslope;
          1613 numeric eventh, eventslope;
          1614 eventh := onelineh;
          1615 eventslope := 0.25;
          1616
   nextra Margins for namespaces in normal (nextra) and inset (niextra) boxes. Inset
   niextra boxes are the ENUM, SELECT and TYPE boxes.
          1617 %%% extra namespaces for boxed names
          1618 newinternal nextra, niextra;
          1619 numeric nextra, niextra;
          1620 nextra := 2u;
          1621 niextra := nextra+sdtbs;
          1622
```

ndextra Margin for namespace on a relationship line.

```
1623 %%% extra namespace for attribute names
1624 newinternal ndextra;
1625 numeric ndextra;
1626 ndextra := nextra+dotdiam;
1627
```

4.2 Utility routines

VH Calculates the intersection point z\$vh between the vertical line through z\$ and the horizontal line through z\$\$ (see Figure 3).

```
1628 %%% calculates mid-point on a path like |_ (vertical, horizontal)
1629 %%% final points are: z$, z$vh, z$$, where z$vh=(x$,y$$)
1630 def VH(suffix $, $$) =
1631 z$vh=(x$,y$$);
1632 enddef;
1633
```

VhV Calculates the point z\$vhv on the vertical line through z\$ and the point z\$\$vhv on the vertical line through z\$\$, such that the line z\$vhv--z\$\$vhv is horizontal and centered vertically between the given points (see Figure 3).

- HvH Calculates the point z\$hvh on the horizontal line through z\$ and the point z\$\$hvh on the horizontal line through z\$\$, such that the line z\$hvh--z\$\$hvh is vertical and centered horizontally between the given points (see Figure 3). 1642 %% Calculates mid-points on a path like -|_ (horizontal, vertical, horizontal) 1643 %% final points are: z\$, z\$hvh, z\$\$hvh, z\$\$ 1644 def HvH(suffix \$, \$\$) = 1645 x\$hvh=1/2[x\$,x\$\$]; y\$hvh=y\$; 1646 z\$\$hvh=(x\$hvh,y\$\$); 1647 enddef; 1648
- VyV A generalisation of VhV. Calculates the point z\$vyv on the vertical line through z\$ and the horizontal line through y0, and the point z\$\$vyv on the vertical line through z\$\$ and the horizontal line through y0 (see Figure 4).

```
1649 %%% Calculates turning points on a U shaped path (vertical, horizontal, vertical)
1650 %%% final points are: z$, z$vyv, z$$vyv, z$$
1651 def VyV(suffix $, @, $$) =
1652 z$$vyv=(x$$, y@);
1653 z$vyv =(x$, y@);
1654 enddef;
1655
```

HxH A generalisation of HvH. Calculates the point z\$hxh on the horizontal line through z\$ and the vertical line through x@, and the point z\$\$hgh on the horizontal line through z\$\$ and the vertical line through x@ (see Figure 4).

```
1656 %%% Calculates corner points rotated U shaped path (horizontal, vertical, horizontal)
1657 %%% final points are: z$, z$hxh, z$$hxh, z$$
1658 def HxH(suffix $, @, $$) =
1659 z$$hxh=(x@,y$$);
1660 z$hxh =(x@,y$);
1661 enddef;
1662
```

```
namespace namespace((name))((margin)) calculates the length of the text string (name)
    plus the (margin) length. The body is enclosed in parentheses<sup>5</sup> so you can do
    c=2namespace...
    1663 %%% calculates length taken up by typesetting str, plus margin
    1664 def namespace(text str)(expr margin) =
```

```
1665 (xpart lrcorner(str) - xpart llcorner(str) + margin)
1666 enddef;
1667
```

dashes Shorthand for a dashed line style. Use as: draw ... dashes;.

1668 def dashes =
1669 dashed evenly
1670 enddef;
1671

⁵Requested by Stephan Hennig, ctt thread [MP, expressg, latexmp] namespace and textext, 16 March 2004

```
dots Shorthand for a dotted line style. Use as: draw ... dots;.
    1672 def dots =
    1673    dashed withdots
    1674 enddef;
    1675
```

dashedgrid dashedgrid($\langle nx \rangle$, $\langle ny \rangle$, $\langle dist \rangle$) draws a thin dashed grid with $\langle nx \rangle$ and $\langle ny \rangle$ divisions in the x and y directions, with $\langle dist \rangle$ between the grid lines. The grid lines are numbered.

```
1676 % draw a general dashed grid
1677 def dashedgrid(expr nx, ny, dist) =
1678
      save zg_, zg__, oldpen;
      pair zg_[], zg__[];
1679
     pen oldpen; oldpen = currentpen;
1680
1681
     pickup thinpen;
1682
     for i = 0 upto nx:
        zg_[i] = (i*dist, 0); zg_[i+1000] = (i*dist, ny*dist);
1683
        draw zg_[i]--zg_[i+1000] dashes;
1684
1685
        label.bot(decimal(i), zg_[i]);
1686
      endfor
1687
      for i = 0 upto ny:
        zg__[i] = (0, i*dist); zg__[i+1000] = (nx*dist, i*dist);
1688
        draw zg__[i]--zg__[i+1000] dashes;
1689
        label.lft(decimal(i), zg__[i]);
1690
1691
      endfor
     pickup oldpen;
1692
1693 enddef;
1694
```

drawgrid draws a thin dashed grid filling the permitted space for an ISO Standard diagram. The units are in mm and the lines are at 10mm intervals with labeling in terms of mm.

```
1695 %%% draw a 1cm spaced grid of 16(x) by 21(y) cms. (units are mm)
1696 def drawgrid =
1697 dashedgrid(16, 21, 10mm);
1698 enddef;
1699
```

The box drawing routines take as an argument the suffix (\$) of the left-hand bottom corner point of the box, and normally the length and height of the box. The coordinates of the box corners are calculated (z\$b1, z\$br, z\$tr and z\$t1). The midpoints of the sides of the box (z\$bm, z\$mr, z\$tm, z\$m1) are calculated as well. The text 'center' of the box is z\$c, while the geometric center of the box will be at the intersection point of the lines z\$bm--z\$tm and z\$m1--z\$mr. See Figure 5 for an illustration.

rectpoints The routine rectpoints(\$, 1, h) calculates the corner and midpoints of a rectangular box with bottom left at z\$, length 1 and height h. Note that it does *not* calculate the center point (see Figure 5).

```
1700 %%% calculates corner and midpoints of a rectangle,
1701 %%% bottom left at $, length 1, height h
1702 def rectpoints(suffix $)(expr 1, h) =
1703 z$bl = z$;
1704 z$tr = (x$+1, y$+h);
1705
     z$br = (x$tr, y$bl);
1706
     z$tl = (x$bl, y$tr);
     z$ml = 1/2[z$bl, z$tl];
1707
1708 z$mr = 1/2[z$br, z$tr];
     z$bm = 1/2[z$bl, z$br];
1709
1710 z$tm = 1/2[z$tl, z$tr];
1711 enddef;
1712
```

rhompoints The routine rhompoints(\$, 1, h, s) calculates the corner and midpoints of a rhomboid length 1, height h, sideslope s, and positioned with its bottom left hand corner at z\$. The center point is *not* calculated (see Figure 6).

```
1713 %%% calculate corner, midpoints and center point of a rhomboid.
1714 def rhompoints(suffix $)(expr 1, h, s) =
1715
     save eshift;
1716 numeric eshift; eshift = s*h;
1717 z$bl = z$;
1718 z$tr = (x$+l+eshift, y$+h);
1719 z$br = (x$bl+1, y$bl);
1720 z$tl = (x$bl+eshift, y$tr);
1721 z$ml = 1/2[z$bl, z$tl];
     z$mr = 1/2[z$br, z$tr];
1722
     z$bm = (1/2[x$ml,x$mr], y$bl);
1723
1724
     z$tm = (x$bm, y$tr);
1725 enddef;
1726
```

circpoints The routine circpoints(\$, d) calculates the 'corner' and 'midpoints' on the circumference of a circle, center z\$ and diameter d.

```
1727 %%% calculate circumferential points on a circle
1728 def circpoints(suffix $)(expr diam) =
1729
     save rad, sinrad, cosrad;
     numeric rad, sinrad, cosrad;
1730
     rad = diam/2;
1731
     sinrad = rad*(sind 45);
1732
1733 cosrad = rad*(cosd 45);
1734 z$c=z$;
1735 z$ml=(x$c-rad, y$c);
1736 z$mr=(x$c+rad, y$c);
1737 z$bm=(x$c, y$c-rad);
1738 z$tm=(x$c, y$c+rad);
     z$tr=(x$c+cosrad, y$c+sinrad);
1739
1740
     z$bl=(x$c-cosrad, y$c-sinrad);
1741 z$br=(x$tr, y$bl);
```

```
1742 z$tl=(x$bl, y$tr);
1743 enddef;
1744
```

4.3 Path routines

The binary operator ~ is a reimplementation of the METAFONT plain base softjoin path connector (METAFONTbook, page 266); it connects paths via a small circular arc radius smoothrad.

```
1745 %%% circular arc join between two paths
1746 tertiarydef p ~ q =
1747
    begingroup
     c_ := fullcircle scaled 2smoothrad shifted point 0 of q;
1748
     a_ := ypart(c_ intersectiontimes p);
1749
     b_ := ypart(c_ intersectiontimes q);
1750
     if a_ < 0: point 0 of p{direction 0 of p} else: subpath(0,a_) of p fi
1751
        ... if b_ < 0: {direction infinity of q}point infinity of q
1752
            else: subpath(b_,infinity) of q fi
1753
1754
     endgroup
1755 \text{ enddef};
1756
```

sharply sharply(zi, zj, ...zn) creates a piecewise linear path through the given points. (The code is based on the flex routine, page 267 of the METAFONTbook).

```
1757 %%% piecewise linear path between the given points
1758 def sharply(text t) = % t is a list of pairs
1759 hide(n_:=0; for z=t: z_[incr n_]:=z; endfor)
1760 z_1 for k=2 upto n_: --z_[k] endfor
1761 enddef;
1762
```

smoothly smoothly(zi, zj, ...zn) creates a piecewise linear path through the given points, with the sharp corners replaced by circular arcs of radius smoothrad. page 267 of the METAFONTbook).

```
1763 %%% piecewise linear path between the given points with smooth corners
1764 def smoothly(text t) = % t is a list of pairs
1765 hide(n_:=0; for z=t: z_[incr n_]:=z; endfor)
1766 (z_1 for k=2 upto n_-1: --z_[k]) ~ (z_[k] endfor --z_[n_])
1767 enddef;
1768
```

4.4 Line end drawing routines

draw0 Draw an open circle, diameter dotdiam at the end of the vector z\$ to z\$\$. Any underlying graphic/text will be hidden. All the line end drawing routines have similar code.

1769 %%% Draw an open circle at end of vector from \$ to \$\$

 $1770 \ def \ drawO(suffix $, $$) =$

Keep everything inside a group and specify the local variables.

```
1771 begingroup
```

```
1772 save v_, c_, l, p;
```

Specify the types of the local variables.

```
1773 pair v_, c_;
```

```
1774 numeric l;
```

```
1775 path p;
```

 v_{-} is the difference between the start and end points of the line. That is, it is the vector from the end point to the start point.

```
1776 v_ := z$-z$$;
```

++ Using METAPOST's Pythagorean addition operator ++, where a++b means xpart $\sqrt{a^2 + b^2}$, we can calculate the length 1 of the line from the x and y dimension ypart (xpart and ypart) of the vector v_.

1777 l := (xpart v_)++(ypart v_); % length of the line

Using mediation involving the ratio of the circle radius to the length of the line, calculate c_{-} , the required position of the center of the circle at the end of the line.

```
1778 c_ := (dotdiam/(21))[z$$,z$];
```

p is the path for drawing the circle.

1779 p := fullcircle scaled dotdiam shifted c_;

Erase anything underneath the circle.

1780 unfill p;

Now we can draw the circle (twice to make sure it shows).

1781 draw p; draw p;

Finish the local group and end.

```
1782 endgroup
1783 enddef;
1784
```

drawD Draw a closed circle, diameter dotdiam at the end of the vector z\$ to z\$\$. Any underlying graphic/text will be hidden.

```
1785 %%% Draw a closed circle at end of vector from $ to $$
1786 \text{ def drawD(suffix $, $$) =}
1787 begingroup
1788
     save v_, c_, l, p;
1789
     pair v_, c_;
1790 numeric l;
1791
     path p;
     v_{-} := z_{-z};
1792
1793 l := (xpart v_)++(ypart v_); % length of the line
     c_ := (dotdiam/(21))[z$$,z$];
1794
     p := fullcircle scaled dotdiam shifted c_;
1795
1796
     fill p;
1797
      endgroup
```

```
1798 enddef;
1799
```

drawOA Draw an open arrowhead, length gal and base width gab, at the end of the vector from z\$ to z\$\$. The same general code pattern is used for non-circular line end styles, and is described here. It is not too different from drawing circular ends, except that the orientation of the line has to be taken into account.

```
1800 %%% draw an open arrowhead at end of vector from $ to $$
1801 def drawOA(suffix $, $$) =
1802
     begingroup
     save v_, c_, v_u, c_t, c_b, l, hb, p;
1803
     pair v_, c_, v_u, c_t, c_b;
1804
1805
     numeric 1, hb;
1806
     path p;
1807
     hb := gab/2;
1808
     v_ := z$-z$$;
1809
     l := (xpart v_)++(ypart v_); % length of the line
     c_ := (gal/(1))[z$$,z$];
1810
                                     % base of arrowhead
```

Calculate the unit vector in the direction of the line.

```
1811 v_u := unitvector v_;
```

Calculate one of the corner points on the base of the triangle by shifting the midbase point in the direction of the normal to the line by half the base width.

1812 c_t := c_ shifted (-hb*(ypart v_u), hb*(xpart v_u));

Specify that the other corner point is symmetrically opposite the first.

```
1813 c_b - c_ = c_ - c_t;
```

Draw the triangular arrowhead.

drawCA Draw a closed (black) arrowhead, length gal and base width gab, at the end of the vector from z\$ to z\$\$.

```
1820 %%% draw a closed arrowhead at end of vector from $ to $$
1821 \text{ def drawCA(suffix $, $$) =}
1822
     begingroup
1823
     save v_, c_, v_u, c_t, c_b, l, hb, p;
1824
     pair v_, c_, v_u, c_t, c_b;
     numeric 1, hb;
1825
1826
     path p;
     hb := gab/2;
1827
     v_ := z$-z$$;
1828
     l := (xpart v_)++(ypart v_); % length of the line
1829
1830
     c_ := (gal/(l))[z$$,z$];
                                    % base of arrowhead
1831 v_u := unitvector v_;
```

```
1832 c_t := c_ shifted (-hb*(ypart v_u), hb*(xpart v_u));
1833 c_b - c_ = c_ - c_t;
1834 p = c_b--z$$--c_t--cycle;
1835 filldraw p;
1836 endgroup
1837 enddef;
1838
```

drawA Draw a simple arrowhead, length gal and base width gab, at the end of the vector from z\$ to z\$\$.

```
1839 %%% draw a simple arrowhead at end of vector from $ to $$
1840 \text{ def drawA}(\text{suffix }, \text{ $$}) =
1841 begingroup
1842 save v_, c_, v_u, c_t, c_b, l, hb, p;
1843 pair v_, c_, v_u, c_t, c_b;
1844 numeric l, hb;
     path p;
1845
1846 hb := gab/2;
1847
     v_ := z$-z$$;
1848 l := (xpart v_)++(ypart v_); % length of the line
1849 c_ := (gal/(l))[z$$,z$];
                                     % base of arrowhead
1850 v_u := unitvector v_;
1851 c_t := c_ shifted (-hb*(ypart v_u), hb*(xpart v_u));
1852 c_b - c_ = c_ - c_t;
1853 p = c_b - z$--c_t;
1854 draw p;
1855 endgroup
1856 enddef;
1857
```

drawF Draw a fanin, length gfl and base width gfb, at the end of the vector from z\$ to
z\$\$.

```
1858 %/% draws a fanin at the end of the vector from $ to $$
1859 \text{ def drawF(suffix $, $$) =}
     begingroup
1860
     save v_, c_, v_u, c_t, c_b, l, hb, p;
1861
1862
     pair v_, c_, v_u, c_t, c_b;
     numeric 1, hb;
1863
1864
     path p;
1865
     hb := gfb/2;
     v_ := z$-z$$;
1866
    l := (xpart v_)++(ypart v_); % length of the line
1867
    c_ := (gfl/(1))[z$$,z$];
                                   % apex of fan
1868
1869 v_u := unitvector v_;
1870 c_t := z$$ shifted (-hb*(ypart v_u), hb*(xpart v_u));
1871
     c_b - z = z + c_t;
1872 p = c_b--c_--c_t;
1873
     draw p;
1874 endgroup
```
```
1875 enddef;
1876
```

4.5 Line drawing routines

```
drawdots Draw a dotted line between the two points z$ and z$$.
              1877 %%% Draw a dotted line from $ to $$
               1878 def drawdots(suffix $, $$) =
               1879 pickup dotspen;
               1880 draw z$--z$$ dots;
               1881 pickup normalpen;
              1882 enddef;
              1883
drawdotsthree Draw a straight dotted line between the three points z$, z0 and z$$.
              1884 %%% Draws the dotted line $--@--$$
              1885 def drawdotsthree(suffix $, @, $$) =
               1886 pickup dotspen;
               1887 draw z$--z@--z$$ dots;
               1888 pickup normalpen;
               1889 enddef;
              1890
  drawdotsfour Draw a straight dotted line between the four points z$, z0, z00 and z$$.
               1891 %%% Draw the dotted line $--@--@@--$$
               1892 def drawdotsfour(suffix $, 0, 00, $$) =
               1893 pickup dotspen;
               1894 draw z$--z@--z@@--z$$ dots;
               1895 pickup normalpen;
              1896 enddef;
              1897
     drawdots0 Draw a dotted line between the two points z$ and z$$, ending in an open circle,
                diameter dotdiam, at z$$.
               1898 %%% Draw a dotted line from $ to $$ with open circle at $$
               1899 \text{ def drawdotsO(suffix $, $$) =}
               1900 pickup dotspen;
               1901 draw z$--z$$ dots;
               1902 pickup normalpen;
               1903 drawO($, $$);
               1904 enddef;
               1905
drawdotsthree0 Draw a straight dotted line between the three points z$, z@ and z$$, ending in an
                open circle, diameter dotdiam, at z$$.
               1906 %%% Draws the dotted line $--@--$$
               1907 def drawdotsthreeO(suffix $, @, $$) =
               1908 pickup dotspen;
               1909 draw z$--z@--z$$ dots;
```

```
1910 pickup normalpen;
              1911 drawO(@, $$);
              1912 \text{ enddef};
             1913
drawdotsfour0 Draw a straight dotted line between the four points z$, z0, z00 and z$$, ending
               in an open circle, diameter dotdiam, at z$$.
              1914 %%% Draw the dotted line $--@--@@--$$
              1915 def drawdotsfourO(suffix $, 0, 00, $$) =
             1916 pickup dotspen;
              1917 draw z$--z@--z$$ dots;
              1918 pickup normalpen;
              1919 drawO(@@, $$);
             1920 enddef;
             1921
   drawdots00 Draw a dotted line between the two points z$ and z$$, ending in open circles,
               diameter dotdiam, at z$ and z$$.
              1922 %%% Draw the dotted line from $ to $$, ending in circles diameter dotdiam at $ and $$
             1923 \text{ def drawdotsOO(suffix $, $$) =}
             1924 pickup dotspen;
              1925 draw z$--z$$ dots;
                   pickup normalpen;
              1926
                   drawO($, $$); drawO($$, $);
              1927
              1928 enddef;
              1929
     drawdash Draw a dashed line between the two points z$ and z$$.
             1930 %%% draws a dashed line from $ to $$
             1931 def drawdash(suffix $, $$) =
              1932 draw z$--z$$ dashes;
             1933 enddef;
             1934
drawdashthree Draw a straight dashed line between the three points z$, z@ and z$$.
             1935 %%% draws the dashed line $--@--$$
              1936 def drawdashthree(suffix $, @, $$) =
              1937 draw z$--z@--z$$ dashes;
             1938 enddef;
             1939
 drawdashfour Draw a straight dashed line between the four points z$, z@, z@@ and z$$.
             1940 %%% draws the dashed line -Q--QQ--
              1941 def drawdashfour(suffix $, 0, 00, $$) =
              1942 draw z$--z@--z$$ dashes;
             1943 enddef;
```

1944

```
drawdash0 Draw a dashed line between the two points z$ and z$$, ending in an open circle,
                 diameter dotdiam, at z$$.
                1945 %%% draws a dashed line from $ to $$, ending in a circle diameter dotdiam at $$
                1946 def drawdashO(suffix $, $$) =
                1947 draw z$--z$$ dashes;
                1948 drawO($, $$);
                1949 enddef;
                1950
 drawdashthree0 Draw a straight dashed line between the three points z$, z@ and z$$, ending in
                 an open circle, diameter dotdiam, at z$$.
                1951 %%% draws the dashed line $--@--$$ with circle at $$
                1952 def drawdashthreeO(suffix $, @, $$) =
                1953
                     draw z$--z@--z$$ dashes;
                1954
                     drawO(@, $$);
                1955 enddef;
                1956
  drawdashfour0 Draw a straight dashed line between the four points z$, z@, z@@ and z$$, ending
                 in an open circle, diameter dotdiam, at z$$.
                1957 %%% draws the dashed line $--Q--QQ--$$ with circle at $$
                1958 \text{ def drawdashfourO(suffix $, 0, 00, $$) =}
                     draw z$--z0--z00--z$$ dashes;
                1959
                     drawO(@@, $$);
                1960
                1961 enddef;
                1962
     drawdash00 Draw a dashed line between the two points z$ and z$$, ending in open circles,
                 diameter dotdiam, at z$ and z$$.
                1963 % drawdash00($, $$)
                1964 %%% draws a dashed line from $ to $$, ending in circles diameter dotdiam at $ and $$
                1965 \text{ def drawdashOO(suffix $, $$) =}
                1966 draw z$--z$$ dashes;
                1967 drawO($, $$); drawO($$, $);
                1968 \text{ enddef};
                1969
     drawnormal Draw a normal thickness line between the two points z$ and z$$.
                1970 %%% draws a normal line from $ to $$.
                1971 def drawnormal(suffix $, $$) =
                1972 draw z$--z$$;
                1973 enddef;
                1974
drawnormalthree Draw a straight normal thickness line between the three points z, z and z, z.
                1975 % drawnormalthree($, @, $$)
                1976 %%% draws the normal line $--@--$$
                1977 def drawnormalthree(suffix $, @, $$) =
                1978 draw z$--z@--z$$;
```

```
75
```

```
1979 enddef;
                1980
  drawnormalfour Draw a straight normal thickness line between the four points z$, z0, z00 and
                 z$$.
                1981 %%% draws the normal line $--@--@@--$$
                1982 def drawnormalfour(suffix , 0, 00, ) =
                1983 draw z$--z@--z$$;
                1984 enddef:
                1985
    drawnormal0 Draw a straight normal thickness line between the two points z$ and z$$, ending
                 in an open circle, diameter dotdiam, at z$$.
                1986 % drawnormalO($, $$)
                1987 %%% draws a normal line from $ to $$, ending in a circle diameter dotdiam at $$
                1988 def drawnormalO(suffix $, $$) =
                1989
                     draw z$--z$$;
                     drawO($, $$);
                1990
                1991 enddef;
                1992
ending in an open circle, diameter dotdiam, at z$$.
                1993 %/% draws the normal line $--@--$$, ending in a circle at $$
                1994 def drawnormalthreeO(suffix $, @, $$) =
                1995 draw z$--z@--z$$;
                1996 drawO(@, $$);
                1997 enddef;
                1998
 drawnormalfour0 Draw a straight normal thickness line between the four points z$, z0, z00 and
                 z$$, ending in an open circle, diameter dotdiam, at z$$.
                1999 %%% draws the line $--@--@@--$$, ending in a circle at $$
                2000 def drawnormalfourO(suffix $, @, @@, $$) =
                     draw z$--z0--z00--z$$;
                2001
                2002 draw0(@@, $$);
                2003 enddef;
                2004
                2005
    drawnormalOO Draw a straight normal thickness line between the two points z$ and z$$, ending
                 in open circles, diameter dotdiam, at z$ and z$$.
                2006 % drawnormalOO($, $$)
                2007 %%% draws a normal line from $ to $$, ending in circles diameter dotdiam at $ and $$
                2008 def drawnormalOO(suffix $, $$) =
                2009
                     draw z$--z$$;
                2010 drawO($, $$); drawO($$, $);
                2011 enddef;
```

76

2012

```
drawnormalD Draw a straight normal thickness line between the two points z$ and z$$, ending
                 in a black dot, diameter dotdiam, at z$$.
                2013 %%% draws a normal line from $ to $$, ending in a dot diameter dotdiam at $$
                2014 def drawnormalD(suffix $, $$) =
                2015 draw z$--z$$;
                2016 drawD($, $$);
                2017 \text{ enddef};
                2018
ending in a black dot, diameter dotdiam, at z$$.
                2019 %%% draws the normal line -Q--$, ending in a dot at $$
                2020 def drawnormalthreeD(suffix $, @, $$) =
                     draw z$--z@--z$$;
                2021
                2022 drawD(@, $$);
                2023 enddef;
                2024
 drawnormalfourD Draw a straight normal thickness line between the four points z$, z0, z00 and
                 z$$, ending in a black dot, diameter dotdiam, at z$$.
                2025 %%% draws the normal line $--@--@@--$$, ending in a dot at $$
                2026 def drawnormalfourD(suffix $, 0, 00, $$) =
                2027
                     draw z$--z0--z00--z$$;
                2028
                     drawD(@@, $$);
                2029 enddef;
                2030
                2031
    drawnormalDD Draw a straight normal thickness line between the two points z$ and z$$, ending
                 in black dots, diameter dotdiam, at z$ and z$$.
                2032 %%% draws a normal line from $ to $$, ending in dots diameter dotdiam at $ and $$
                2033 def drawnormalDD(suffix $, $$) =
                2034 draw z$--z$$;
                2035 drawD($, $$); drawD($$, $);
                2036 enddef;
                2037
    drawnormalOA Draw a straight normal thickness line between the two points z$ and z$$, ending
                 with an open arrowhead, length gal and base width gab, at z$$.
                2038 %/% draws a normal line from $ to $$, ending with an open arrowhead at $$
                2039 def drawnormalOA(suffix $, $$) =
                     draw z$--z$$:
                2040
```

```
2041 drawOA($, $$);
2042 enddef;
```

- 2043
- drawnormalCA Draw a straight normal thickness line between the two points z\$ and z\$\$, ending with a closed (black) arrowhead, length gal and base width gab, at z\$\$.

```
2044 %%% draws a normal line from $ to $$, ending with a closed arrowhead at $$
                 2045 def drawnormalCA(suffix $, $$) =
                      draw z$--z$$;
                 2046
                      drawCA($, $$);
                 2047
                 2048 enddef;
                 2049
ending in a closed arrowhead at z$$.
                 2050 %%% draws the normal line $--@--$$, ending in a black arrowhead at $$
                 2051 def drawnormalthreeCA(suffix $, @, $$) =
                 2052 draw z$--z@--z$$;
                     drawCA(@, $$);
                 2053
                 2054 enddef;
                 2055
 drawnormalfourCA Draw a straight normal thickness line between the four points z$, z@, z@@ and
                  z$$, ending in a closed arrowhead at z$$.
                 2056 %%% draws the normal line $--@--@@--$$, ending in a black arrowhead at $$
                 2057 def drawnormalfourCA(suffix , 0, 00, 
                      draw z$--z0--z00--z$$;
                 2058
                 2059
                      drawCA(@@, $$);
                 2060 enddef;
                 2061
     drawnormal F Draw a straight normal thickness line between the two points z$ and z$$, ending
                  with a fanin, length gfl and base width gfb, at z$$.
                 2062 %%% draws a normal line from $ to $$, ending with a fanin at $$
                 2063 def drawnormalF(suffix $, $$) =
                 2064 draw z$--z$$;
                 2065 drawF($, $$);
                 2066 \text{ enddef};
                 2067
     drawnormalFO Draw a straight normal thickness line between the two points z$ and z$$, ending
                  with a fanin, length gfl and base width gfb, at z$ and an open circle, diameter
                  dotdiam, at z$$.
                 2068 %%% draws a normal line from $ to $$, with a fanin at $ and an open circle at $$
                 2069 def drawnormalFO(suffix $, $$) =
                 2070 draw z$--z$$;
                      drawO($, $$); drawF($$, $);
                 2071
                 2072 \text{ enddef}:
                 2073
        drawthick Draw a straight thick line between the two points z$ and z$$
                 2074 %%% draws a thick line from $ to $$.
                 2075 def drawthick(suffix $, $$) =
```

2076 pickup thickpen; 2077 draw z\$--z\$\$;

```
2078 pickup normalpen;
2079 enddef;
2080
```

drawthick0 Draw a straight thick line between the two points z\$ and z\$\$, ending with an open circle, diameter dotdiam, at z\$\$. 2081 %%% draws a thick line from \$ to \$\$, ending in a circle diameter dotdiam at \$\$ 2082 def drawthick0(suffix \$, \$\$) =

```
2083 pickup thickpen;
2084 draw z$--z$$;
2085 draw0($, $$);
2086 pickup normalpen;
2087 enddef;
2088
```

drawthick00 Draw a straight thick line between the two points z\$ and z\$\$, ending with open circles, diameter dotdiam, at z\$ and z\$\$.

```
2089 %%% draws a thick line from $ to $$, ending in circles diameter dotdiam at $ and $$
2090 def drawthickOO(suffix $, $$) =
2091 pickup thickpen;
2092 draw z$--z$$;
2093 drawO($, $$); drawO($$, $);
2094 pickup normalpen;
2095 enddef;
2096
```

smooth Rounds the join at z⁰ between the two straight lines z^{\$--z⁰--z^{\$}. The radius of the circular arc is smoothrad. The code employs the same technique as used for putting dots, etc., at the end of a line. However, in this case we 'undraw' the sharp corner before drawing the arc.}

```
2097 %%% replaces the sharp corner on $--@--$$ with a circular arc radius smoothrad
2098 \text{ def smooth}(\text{suffix $, 0, $$}) =
     begingroup
2099
     save v_, c_, l, p;
2100
     pair v_, v_', c_, c_';
2101
2102
     path p;
2103
     v_ := z@-z$;
2104 l := (xpart v_)++(ypart v_);
                                       % length of $--@
2105 c_ := (smoothrad/l)[z@,z$];
                                        % start of arc on $--@
2106 v_' := z$$-z@;
2107 l := (xpart v_')++(ypart v_');
                                       % length of @--$$
2108 c_' := (smoothrad/l)[z@,z$$];
                                        % end of arc on @--$$
2109 undraw c_--z@--c_';
                                        % blank original join
     draw c_{v_}..{v_'}c_';
                                        % draw the arc
2110
      endgroup
2111
2112 enddef;
2113
```

smoothtwo Rounds the joins at z0 and z00 between the three straight lines z\$--z0--z00--z\$\$. The radius of the circular arc is smoothrad.

```
2114 %/% replaces the sharp corners on $--@--@@--$$ with a circular arc radius smoothrad
2115 def smoothtwo(suffix $, @, @@, $$) =
2116 smooth($, @, @@); smooth(@, @@, $$);
2117 enddef;
2118
```

```
smoothdash Rounds the join at z@ between the two dashed straight lines z$--z@--z$$. The
radius of the circular arc is smoothrad.
```

```
2119 %%% replaces the sharp corner on the dashed lines $--@--$$
2120 %%% with a circular arc radius smoothrad
2121 def smoothdash(suffix $, @, $$) =
2122 begingroup
2123 save v_, c_, l, p;
2124 pair v_, v_', c_, c_';
2125 path p;
2126 v_ := z@-z$;
2127 l := (xpart v_)++(ypart v_);
                                      % length of $--@
2128 c_ := (smoothrad/l)[z0,z$];
                                      % start of arc on $--@
2129 v_' := z$$-z@;
2130 l := (xpart v_')++(ypart v_');
                                      % length of @--$$
2131 c_' := (smoothrad/l)[z@,z$$];
                                      % end of arc on @--$$
2132 undraw c_--z@--c_';
                                      % blank original join
2133 draw c_{v_}..{v_'}c_' dashes;
                                      % draw the dashed arc
2134 endgroup
2135 enddef;
2136
```

```
smoothdots Rounds the join at z<sup>0</sup> between the two dotted straight lines z<sup>$--z<sup>0</sup>--z$$</sup>. The
radius of the circular arc is smoothrad.
```

```
2137 %%% replaces the sharp corner on the dotted line $--@--$$
2138 %%% with a circular arc radius smoothrad
2139 def smoothdots(suffix $, @, $$) =
2140 begingroup
2141
     save oldpen;
     pen oldpen; oldpen := currentpen;
2142
2143
     save v_, c_, l, p;
2144 pair v_, v_', c_, c_';
2145 path p;
2146 v_ := z@-z$;
2147 l := (xpart v_)++(ypart v_);
                                      % length of $--@
2148 c_ := (smoothrad/l)[z@,z$];
                                      \% start of arc on $--0
2149 v_' := z$$-z@;
2150 l := (xpart v_')++(ypart v_');
                                      % length of @--$$
2151 c_' := (smoothrad/l)[z@,z$$];
                                      \% end of arc on Q--$$
2152 undraw c_--z@--c_';
                                      % blank original join
2153 pickup dotspen;
2154 draw c_{v_}..{v_'}c_' dots;
                                      % draw the dotted arc
2155 pickup oldpen;
2156 endgroup
```

2157 enddef; 2158

4.6 Box drawing routines

The box drawing routines take as an argument the suffix (\$) of the left-hand bottom corner point of the box, and normally the length and height of the box. There is normally also a text argument that gets typeset at the 'center' of the box. The coordinates of the box corners are calculated (z\$b1, z\$br, z\$tr and z\$t1). The midpoints of the sides of the box (z\$bm, z\$mr, z\$tm, z\$m1) are calculated as well. The text 'center' of the box is z\$c, while the geometric center of the box will be at the intersection point of the lines z\$bm--z\$tm and z\$m1--z\$mr. See Figure 5 for an illustration.

```
drawSCHEMA drawSCHEMA(\langle suffix \rangle, \langle l \rangle, \langle h \rangle)(\langle name \rangle) draws a SCHEMA box.
```

```
2159 %%% draws a schema box, bottom left at $, length 1, height h
2160 def drawSCHEMA(suffix $)(expr 1, h)(text str) =
     rectpoints($, 1, h);
2161
2162 x$c = 1/2[x$ml, x$mr];
2163
     y$c = 1/2[y$ml, y$tl];
     draw z$bl--z$br--z$tl--cycle; % outer box
2164
     draw z$ml--z$mr;
                                           % dividing line
2165
     label(str, z$c);
2166
2167 enddef;
2168
```

drawSDT drawSDT($\langle suffix \rangle$)($\langle name \rangle$) draws a simple data type box of length sdtbl and height sdtbh.

```
2169 % drawSDT($)(name)
2170 % draw a simple data type box, bottom left at $
2171 def drawSDT(suffix $)(text str) =
2172 rectpoints($, sdtbl, sdtbh);
     z$ti = (x$tr-sdtbs, y$tr);
2173
     z$bi = (x$ti, y$br);
2174
     z$c = 1/2[z$bl,z$ti];
2175
     draw z$bl--z$br--z$tr--z$tl--cycle;
2176
     draw z$bi--z$ti;
2177
2178 label(str, z$c);
2179 enddef;
2180
```

drawASDT drawASDT ($\langle suffix \rangle, \langle l \rangle, \langle h \rangle$) ($\langle name \rangle$) draws a simple data type box.

```
2181 % drawASDT($, 1, h)(name)
2182 % draw a simple data type box, bottom left at $, length 1, height h
2183 def drawASDT(suffix $)(expr 1, h)(text str) =
2184 rectpoints($, 1, h);
2185 z$ti = (x$tr-sdtbs, y$tr);
2186 z$bi = (x$ti, y$br);
```

```
2187 z$c = 1/2[z$bl,z$ti];
2188 draw z$bl--z$br--z$tr--z$tl--cycle;
2189 draw z$bi--z$ti;
2190 label(str, z$c);
2191 enddef;
2192
```

The drawNAME($\langle suffix \rangle$) routines draw a simple data type box of the given name using the applicable lengths and heights.

```
drawBINARY
```

```
drawBOOLEAN 2193 def drawBINARY(suffix $) = drawASDT($)(sdtbl, sdtbh)("BINARY");
   drawCOMPLEX 2194 enddef:
drawEXPRESSION 2195 def drawBOOLEAN(suffix $) = drawASDT($)(sdtbl, sdtbh)("BOOLEAN");
               2196 enddef;
               2197 def drawCOMPLEX(suffix $) = drawASDT($)(sdtbl, sdtbh)("COMPLEX");
               2198 enddef;
               2199 def drawEXPRESSION(suffix $) = drawASDT($)(sdtbel, sdtbeh)("EXPRESSION");
               2200 enddef;
   drawGENERIC
   drawINTEGER 2201 def drawGENERIC(suffix $) = drawASDT($)(sdtbl, sdtbh)("GENERIC");
   drawLOGICAL 2202 enddef;
    drawNUMBER 2203 def drawINTEGER(suffix $) = drawASDT($)(sdtbl, sdtbh)("INTEGER");
               2204 enddef:
               2205 def drawLOGICAL(suffix $) = drawASDT($)(sdtbl, sdtbh)("LOGICAL");
               2206 enddef;
               2207 def drawNUMBER(suffix $) = drawASDT($)(sdtbl, sdtbh)("NUMBER");
               2208 enddef;
      drawREAL
    drawSTRING 2209 def drawREAL(suffix $) =
                                                 drawASDT($)(sdtbl, sdtbh)("REAL");
               2210 enddef:
               2211 def drawSTRING(suffix $) = drawASDT($)(sdtbl, sdtbh)("STRING");
               2212
                     enddef;
               2213
      drawENUM drawENUM(\langle suffix \rangle, \langle l \rangle, \langle h \rangle) (\langle name \rangle) draws an ENUMERATION type box.
               2214 % drawENUM($, 1, h)(name)
               2215\ \mbox{\ensuremath{\sc km}\sc km}\ \mbox{\sc h} draw an enumeration type box, bottom left at $, length 1, height h
               2216 def drawENUM(suffix $)(expr 1, h)(text str) =
               2217 rectpoints($, 1, h);
               2218 z$ti = (x$tr-sdtbs, y$tr);
               2219 z$bi = (x$ti, y$br);
               2220 z$c = 1/2[z$bl,z$ti];
               2221 draw z$bl--z$br--z$tl--cycle dashes;
               2222 draw z$bi--z$ti dashes;
               2223 label(str, z$c);
               2224 enddef;
               2225
```

```
drawSELECT drawSELECT (\langle suffix \rangle, \langle l \rangle, \langle h \rangle) (\langle name \rangle) draws a SELECT type box.
           2226 % drawSELECT($, 1, h)(name)
           2227 %/% draw a select type box, bottom left at $, length 1, height h
           2228 def drawSELECT(suffix $)(expr 1, h)(text str) =
           2229 rectpoints($, 1, h);
           2230 z$ti = (x$tl+sdtbs, y$tl);
           2231 z$bi = (x$ti, y$bl);
           2232 z$c = 1/2[z$br,z$ti];
           2233 draw z$bl--z$br--z$tl--cycle dashes;
           2234 draw z$bi--z$ti dashes;
           2235 label(str, z$c);
           2236 enddef;
           2237
  drawTYPE drawTYPE (\langle suffix \rangle, \langle l \rangle, \langle h \rangle) (\langle name \rangle) draws a user-defined TYPE type box.
           2238 % drawTYPE($, 1, h)(name)
           2239 %%% draw a simple user defined TYPE box, bottom left at $, length 1, height h
           2240 def drawTYPE(suffix $)(expr 1, h)(text str) =
           2241 rectpoints($, 1, h);
           2242 z$c = 1/2[z$bl,z$tr];
           2243 draw z$bl--z$br--z$tl--cycle dashes;
           2244 label(str, z$c);
           2245 \text{ enddef};
           2246
   drawENT drawENT(\langle suffix \rangle, \langle l \rangle, \langle h \rangle)(\langle name \rangle) draws an ENTITY box.
           2247 % drawENT($, 1, h)(name)
           2248 %/% draw an entity box, bottom left at $, length 1, height h
           2249 def drawENT(suffix $)(expr 1, h)(text str) =
           2250 rectpoints($, 1, h);
           2251 z$c = 1/2[z$bl,z$tr];
           2252 draw z$bl--z$br--z$tr--z$tl--cycle;
           2253 label(str, z$c);
           2254 \text{ enddef};
           2255
    drawOB drawOB(\langle suffix \rangle, \langle l \rangle, \langle h \rangle) draws a rectangular box with rounded corners (an
             EXPRESS-G oval box).
           2256 % drawOB($, 1, h)
           2257 % draw an oval box, bottom left at $, length 1, height h
           2258 def drawOB(suffix $)(expr 1, h) =
           2259 save rad;
           2260 numeric rad; rad := h/2;
           2261 rectpoints($, 1, h);
           2262 z$cl = 1/2[z$bl, z$tl];
           2263 z$cr = 1/2[z$br, z$tr];
           2264 z$bli = (x$bl+rad, y$bl);
           2265 z$tli = (x$bli, y$tl);
```

```
2266 z$bri = (x$br-rad, y$br);
```

```
2267 z$tri = (x$bri, y$tr);
         2268 z$c = 1/2[z$bl,z$tr];
               draw z$bli--z$bri..z$cr..z$tri--z$tli..z$cl..cycle;
         2269
         2270 \text{ enddef};
         2271
drawPREF drawPREF (\langle suffix \rangle, \langle l \rangle, \langle h \rangle) (\langle name \rangle) draws a page reference oval box.
         2272 % drawPREF($, 1, h)(name)
         2273 %/% draw a page reference box oval, bottom left at $, length 1, height h
         2274 def drawPREF(suffix $)(expr 1, h)(text str) =
         2275 drawOB($, 1, h);
         2276 label(str, z$c);
         2277 enddef;
         2278
 drawISU drawISU(\langle suffix \rangle, \langle l \rangle, \langle h \rangle) (\langle name \rangle) draws an interschema USE box.
         2279 % drawISU($, 1, h)(name)
         2280 %/% draw an interschema USE box, bottom left at $, length 1, height h
         2281 def drawISU(suffix $)(expr 1, h)(text str) =
         2282 save quarter;
         2283 numeric quarter; quarter := h/4;
         2284 rectpoints($, 1, h);
         2285 z$o = (x$, y$+quarter);
         2286 drawOB($o, 1, 2quarter);
         2287 z$c = 1/2[z$bl,z$tr];
         2288 label(str, z$c);
         2289 draw z$bl--z$br--z$tr--z$tl--cycle;
         2290 \text{ enddef}:
         2291
drawISUR drawISUR(\langle suffix \rangle, \langle l \rangle, \langle h \rangle) (\langle name \rangle) (\langle rename \rangle) draws an interschema USE RE-
           NAME box.
         2292 % drawISUR($, 1, h)(name)(rename)
         2293 %%% draw an interschema USE RENAME box, bottom left at $, length 1, height h
         2294 def drawISUR(suffix $)(expr 1, h)(text str, rname) =
         2295 save third;
               numeric third; third := h/3;
         2296
               rectpoints($, 1, h);
         2297
         2298 z$o = (x$, y$+third);
               drawOB($o, 1, third);
         2299
         2300 z$c = 1/2[z$bl, z$tr];
         2301 label(str, z$c);
         2302 z$rnm = 1/2[z$bl,(x$br,y$br+third)];
         2303 draw z$bl--z$br--z$tr--z$tl--cycle;
         2304 label(rname, z$rnm);
         2305 enddef;
         2306
```

drawISR drawISR ($\langle suffix \rangle$, $\langle l \rangle$, $\langle h \rangle$) ($\langle name \rangle$) draws an interschema RERERENCE box.

```
2307 % drawISR($, 1, h)(name)
           2308 %%% draw an interschema REFERENCE box, bottom left at $, length 1, height h
           2309 def drawISR(suffix $)(expr 1, h)(text str) =
           2310 save quarter;
           2311 numeric quarter; quarter := h/4;
           2312 rectpoints($, 1, h);
           2313 z$o = (x$, y$+quarter);
           2314 drawOB($o, 1, 2quarter);
           2315 z$c = 1/2[z$bl,z$tr];
           2316 label(str, z$c);
           2317 draw z$bl--z$tr--z$tl--cycle dashes;
           2318 enddef;
           2319
  drawISRR drawISRR (\langle suffix \rangle, \langle l \rangle, \langle h \rangle) (\langle name \rangle) (\langle rename \rangle) draws an interschema REFER-
             ENCE RENAME box.
           2320 % drawISRR($, 1, h)(name)(rename)
           2321 %%% draw an interschema REFERENCE RENAME box, bottom left at $, length 1, height h
           2322 def drawISRR(suffix $)(expr 1, h)(text str, rname) =
           2323 save third;
           2324 numeric third; third := h/3;
           2325 rectpoints($, 1, h);
           2326 z$o = (x$, y$+third);
           2327 drawOB($o, 1, third);
           2328 z$c = 1/2[z$bl,z$tr];
           2329 label(str, z$c);
           2330 z$rnm = 1/2[z$bl,(x$br,y$br+third)];
           2331 draw z$bl--z$br--z$tr--z$tl--cycle dashes;
           2332 label(rname, z$rnm);
           2333 enddef;
           2334
drawLEVENT drawLEVENT(\langle suffix \rangle, \langle l \rangle, \langle h \rangle)(\langle name \rangle) draws a Local EVENT box, with side
             slope eventslope.
           2335 % drawLEVENT($, 1, h)(name)
           2336 %%% draw a Local EVENT box, bottom left at $, length 1, height h
           2337 def drawLEVENT(suffix $)(expr 1, h)(text str) =
           2338 rhompoints($, 1, h, eventslope);
           2339
                 z = 1/2[zml,zmr];
           2340
                 draw z$bl--z$br--z$tr--z$tl--cycle;
                 label(str, z$c);
           2341
           2342 \text{ enddef};
           2343
drawGEVENT drawGEVENT (\langle suffix \rangle, \langle l \rangle, \langle h \rangle) (\langle name \rangle) draws a Global EVENT box, with side
             slope eventslope.
           2344 % drawGEVENT($, 1, h)(name)
           2345 %%% draw a Global EVENT box, bottom left at $, length 1, height h
           2346 def drawGEVENT(suffix $)(expr 1, h)(text str) =
```

```
rhompoints($, 1, h, eventslope);
               2347
               2348
                     z$c = 1/2[z$ml,z$mr];
                     pickup thickpen;
               2349
                     draw z$bl--z$br--z$tr--z$tl--cycle;
               2350
                     pickup normalpen;
               2351
               2352
                     label(str, z$c);
               2353 enddef;
               2354
drawcirclebox (\langle suffix \rangle, \langle diam \rangle) (\langle name \rangle) draws a circle, center zsuffix and
                diameter \langle diam \rangle, around \langle name \rangle.
               2355 % drawcirclebox($, diam)(name)
               2356 %%% draw a circled name, diameter diam centered at $
               2357 def drawcirclebox(suffix $)(expr diam)(text str) =
                     circpoints($, diam);
               2358
                     draw z$bl..z$bm..z$br..z$mr..z$tr..z$tm..z$tl..z$ml..cycle;
               2359
                     label(str, z$c);
               2360
               2361 enddef;
               2362
```

4.7 Extra BLA variables and routines

Some extra facilities are provided for assistance in drawing non-EXPRESS-G BLA diagrams, such as flow charts, IDEF diagrams or UML structure diagrams.

gdl gdl is the length of a diamond line end style and gdb is the base width.

```
gdb 2363 %%% length and base width of diamond line end styles
2364 newinternal gdl, gdb;
2365 numeric gdl, gdb;
2366 gdl := 2defaultgal;
2367 gdb := 0.75defaultgab;
2368
```

ellipsepoints Calculates the points on an ellipse, center at z\$ with horizontal diameter 1 and vertical diameter h. It does *not* calculate the center point.

```
2369 def ellipsepoints(suffix $)(expr 1, h) =
2370 save epp, move;
     path epp;
2371
2372
     pair move;
     z$ml=(x$-1/2, y$); z$mr=(x$ml+1,y$);
2373
2374 z$tm=(x$, y$+h/2); z$bm=(x$, y$tm-h);
2375 move = 1/2[z$ml,z$mr];
2376 epp = fullcircle scaled h xscaled (1/h) shifted move;
2377 z$tr = point 1.2 of epp;
2378 z$tl = point 2.8 of epp;
     z$bl = point 5.2 of epp;
2379
2380 z$br = point 6.8 of epp;
2381 enddef;
2382
```

```
drawDCA Draw double closed arrowheads, each length gal and base width gab, at end of
         vector from z$ to z$$.
       2383 %%% draws double closed arrowheads at end of vector from $ to $$
       2384 def drawDCA(suffix $, $$) =
       2385
            begingroup
       2386
            save v_, c_, v_u, c_t, c_b, l, hb, p;
       2387 pair v_, c_, v_u, c_t, c_b;
       2388 path p[];
       2389 numeric 1, hb;
       2390 hb := gab/2;
       2391 v_ := z$-z$$;
       2392 l := (xpart v_)++(ypart v_); % length of the line
                                           % base of arrowhead
       2393
            c_ := (gal/(1))[z$$,z$];
            v_u := unitvector v_;
       2394
       2395
            c_t := c_ shifted (-hb*(ypart v_u), hb*(xpart v_u));
       2396
            c_b - c_ = c_ - c_t;
       2397
             p1 := c_b--z$$--c_t--cycle;
       2398
            filldraw p1;
            p2 := p1 shifted ((xpart c_ - x$$), (ypart c_ - y$$));
       2399
       2400
            filldraw p2;
       2401
             endgroup
       2402 enddef;
       2403
 drawOD Draw an open diamond, length gdl and base width gdb, at end of vector from z$
         to z$$.
       2404 %%% draws an open diamond at end of vector from $ to $$
       2405 \text{ def drawOD(suffix $, $$) =}
       2406 begingroup
       2407 save v_, c_, c__, v_u, c_t, c_b, l, hb, p;
       2408 pair v_, c_, c__, v_u, c_t, c_b;
       2409 numeric l, hb;
       2410
            path p;
       2411 hb := gdb/2;
       2412
            v_ := z$-z$$;
            l := (xpart v_)++(ypart v_); % length of the line
       2413
       2414 c_ := (gdl/(21))[z$$,z$];
                                           % base of diamond
       2415 c__ := (gdl/(l))[z$$,z$];
                                           % interior tip of diamond
       2416 v_u := unitvector v_;
       2417 c_t := c_ shifted (-hb*(ypart v_u), hb*(xpart v_u));
       2418 c_b - c_ = c_ - c_t;
       2419 p = c_b--z$$--c_t--c__--cycle;
       2420 unfill p;
       2421
            draw p; draw p;
       2422
            endgroup
       2423\;\texttt{enddef} ;
       2424
```

drawCD Draw a closed diamond, length gdl and base width gdb, at end of vector from z\$

```
to z$$.
2425 %%% draws a closed diamond at end of vector from $ to $$
2426 \text{ def drawCD(suffix $, $$) =}
2427
     begingroup
2428
      save v_, c_, c__, v_u, c_t, c_b, l, hb, p;
2429
      pair v_, c_, c__, v_u, c_t, c_b;
2430
     numeric 1, hb;
2431
     path p;
2432 hb := gdb/2;
2433 v_ := z$-z$$;
2434 l := (xpart v_)++(ypart v_); % length of the line
2435 c_ := (gdl/(21))[z$$,z$];
                                    % base of diamond
2436 c__ := (gdl/(l))[z$$,z$];
                                    % interior tip of diamond
2437 v_u := unitvector v_;
2438 c_t := c_ shifted (-hb*(ypart v_u), hb*(xpart v_u));
2439 c_b - c_ = c_ - c_t;
     p = c_b - z - c_t - c_- - cycle;
2440
2441
     filldraw p;
2442
      endgroup
2443 enddef;
2444
```

drawdashA Draw a straight dashed line between the two points z\$ and z\$\$, ending with an arrowhead, length gal and base width gab, at z\$\$.

```
2445 %%% draws a dashed line from $ to $$, ending with an arrowhead at $$
2446 def drawdashA(suffix $, $$) =
2447   draw z$--z$$ dashes;
2448   drawA($, $$);
2449 enddef;
2450
```

drawdashOA Draw a straight dashed line between the two points z\$ and z\$\$, ending with an open arrowhead, length gal and base width gab, at z\$\$.

```
2451 %%% draws a dashed line from $ to $$, ending with an open arrowhead at $$
2452 def drawdashOA(suffix $, $$) =
2453  draw z$--z$$ dashes;
2454  drawOA($, $$);
2455 enddef;
2456
```

drawnormalDCA Draw a straight normal line between the two points z\$ and z\$\$, ending with double closed arrowheads, each length gal and base width gab, at z\$\$.

```
2457 %%% draws a normal line from $ to $$, ending with double closed arrowheads at $$
2458 def drawnormalDCA(suffix $, $$) =
2459 draw z$--z$$;
2460 drawDCA($, $$);
2461 enddef;
2462
```

```
drawnormalOD Draw a straight normal thickness line between the two points z$ and z$$, ending
                with an open diamond, length gdl and base width gdb, at z$$.
               2463 %%% draws a normal line from $ to $$, ending with an open diamond at $$
               2464 def drawnormalOD(suffix $, $$) =
               2465 draw z$--z$$;
               2466
                     drawOD($, $$);
               2467 \text{ enddef};
               2468
  drawnormalCD Draw a straight normal thickness line between the two points z$ and z$$, ending
                with a closed diamond, length gdl and base width gdb, at z$$.
               2469 %%% draws a normal line from $ to $$, ending with a closed diamond at $$
               2470 def drawnormalCD(suffix $, $$) =
               2471
                     draw z$--z$$;
               2472
                     drawCD($, $$);
               2473 \text{ enddef};
               2474
drawdashcircle Draw a normal thickness dashed open circle, center z$, diameter diam.
               2475 %%% draws an open dashed circle, center z$, diameter diam
               2476 def drawdashcircle(suffix $)(expr diam) =
               2477 circpoints($, diam);
               2478
                    draw z$bl..z$bm..z$br..z$mr..z$tr..z$tm..z$tl..z$ml..cycle dashes;
               2479 enddef;
               2480
   drawcircleA Draw a normal thickness open circle, center z$, diameter diam, and with a coun-
                terclockwise pointing arrow at the topmost point. The arrow has length gal and
                base width gab.
               2481 %%% draws an open circle, center z$, diameter diam, with an arrow at the top
               2482 def drawcircleA(suffix $)(expr diam) =
               2483
                    circpoints($, diam);
                     begingroup
               2484
               2485
                    save c_, c_t, c_b, hb;
               2486 pair c_, c_t, c_b;
                    numeric hb;
               2487
                    hb := gab/2;
               2488
                    c_ := z$tm shifted (gal*right); % base of arrowhead
               2489
                    c_t := c_ shifted (hb*up);
               2490
               2491
                    c_b := c_ shifted (hb*down);
               2492
                    draw z$bl..z$bm..z$br..z$mr..z$tr..z$tm..z$tl..z$ml..cycle;
               2493
                    draw c_t--z$tm--c_b;
               2494
                     endgroup
               2495 \text{ enddef};
               2496
```

drawDot Draw a black dot, center z\$, diameter diam.

```
2497 %%% draws black dot, center z$, diameter diam
2498 def drawDot(suffix $)(expr diam) =
```

```
begingroup
2499
2500
        save p;
2501
        path p;
        p := fullcircle scaled diam shifted z$;
2502
2503
        filldraw p;
2504
      endgroup
2505 \text{ enddef};
2506
2507 %
         \nd{macrocode}
2508 % \end{routine}
2509 %
2510 % \begin{routine}{drawCircledDot}
2511 % Draw a black dot, center |z$|, surrounded by a circle, overall
2512 % diameter |diam|.
2513 % \changes{v1.6}{2004/02/29}{Added drawCircledDot}
2514 %
         \begin{macrocode}
2515 %/% draws black dot surrounded by a circle, center z$, diameter diam
2516 def drawCircledDot(suffix $)(expr diam) =
2517
     begingroup
2518
        save l_, p;
        numeric l_;
2519
        path p[];
2520
2521
        1_ := 5/7diam;
        p1 := fullcircle scaled diam shifted z$;
2522
2523
        unfill p1;
2524
        draw p1;
2525
        p2 := fullcircle scaled l_ shifted z$;
2526
        filldraw p2;
2527
      endgroup
2528 enddef;
2529
2530 %
         \nd{macrocode}
2531 % \end{routine}
2532 %
2533 %
2534 % \begin{routine}{drawcardbox}
2535 % \changes{v1.1}{1999/10/30}{Added drawcardbox routine}
2536 % Draw a rectangular box that has its top righthand corner folded down.
2537 % |m| is the height/length of the fold.
2538 %
         \begin{macrocode}
2539 def drawcardbox(suffix $)(expr 1, h, m)(text str) =
2540 rectpoints($, 1, h);
2541
     begingroup
2542 save c;
2543 pair c[];
2544 c1 = (x$tr-m, y$tr);
2545 c3 = (x$tr, y$tr-m);
2546 c2 = (xpart c1, ypart c3);
2547 draw z$bl--z$br--c3--c1--z$tl--cycle;
2548 draw c1--c2--c3;
```

```
2549 endgroup;
2550 z$c = 1/2[z$ml,z$mr];
2551 label(str, z$c);
2552 enddef;
2553
```

drawdiamondbox Draw a diamond shaped box. Like circles, the box is located at its center point and not at the bottom left corner (see Figure 8).

```
2554 def drawdiamondbox(suffix $)(expr 1, h)(text str) =
2555
      z$ml=(x$-1/2, y$); z$tm=(x$, y$+h/2);
      z$mr=(x$ml+1, y$); z$bm=(x$, y$tm-h);
2556
2557
      z$bl=1/2[z$ml,z$bm];
2558
      z$br=1/2[z$bm,z$mr];
      z$tr=1/2[z$mr,z$tm];
2559
     z$tl=1/2[z$tm,z$ml];
2560
2561
      z;
      draw z$ml--z$bm--z$mr--z$tm--cycle;
2562
      label(str, z$c);
2563
2564 \text{ enddef};
2565
```

drawtwodiamondbox Draw a diamond shaped box with a smaller diamond inside.

```
2566 def drawtwodiamondbox(suffix $)(expr 1, h, mrg)(text str) =
      z$ml=(x$-1/2, y$); z$tm=(x$, y$+h/2);
2567
2568
      z$mr=(x$ml+1, y$); z$bm=(x$, y$tm-h);
2569
     z$bl=1/2[z$ml,z$bm];
     z$br=1/2[z$bm,z$mr];
2570
     z$tr=1/2[z$mr,z$tm];
2571
2572
     z$tl=1/2[z$tm,z$ml];
2573
      z$c=z$;
      begingroup
2574
2575
      save v_, p, tl, sf;
      pair v_;
2576
     numeric tl, sf;
2577
2578
      path p[];
     p1 = z$ml--z$bm--z$mr--z$tm--cycle;
2579
      draw p1;
2580
2581
      v_{-} := z c_{-} z t_{;}
      tl := (xpart v_)++(ypart v_);
2582
      sf = 1.0 - mrg/tl;
2583
      p2 = p1 shifted -z$c scaled sf shifted z$c;
2584
2585
      draw p2;
2586
      endgroup;
      label(str, z$c);
2587
2588
2589 enddef;
2590
```

drawdoublerectangle Draw a double box where tf is the fraction of the height of the top portion. The

ends of the dividing line are z\$tfl and z\$tfr. The text centers are z\$ct and z\$cb for the top and bottom portions.

```
2591 def drawdoublerectangle(suffix $)(expr l, h, tf) =
2592 rectpoints($, l, h);
2593 z$tfl=tf[z$tl,z$bl]; z$tfr=tf[z$tr,z$br];
2594 z$cb=1/2[z$bl,z$tfr];
2595 z$ct=1/2[z$tfl,z$tr];
2596 draw z$bl--z$br--z$tr--z$tl--cycle;
2597 draw z$tfl--z$tfr;
2598 enddef;
2599
```

drawtriplerectangle Draws a triple box, where tf is the fraction of the height for the top portion and bf is the fraction of the height for the bottom portion. Text centers for the three portions are z\$ct, z\$cm and z\$cb for the top, middle and bottom portions. The points at the ends of the dividing lines are z\$tfl and z\$tfr for the top portion and z\$bfl and z\$bfr for the bottom.

```
2600 def drawtriplerectangle(suffix $)(expr 1, h, tf, bf) =
     rectpoints($, 1, h);
2601
     z$tfl=tf[z$tl,z$bl]; z$tfr=tf[z$tr,z$br];
2602
     z$bfl=bf[z$bl,z$tl]; z$bfr=bf[z$br,z$tr];
2603
2604
     z$cb=1/2[z$bl,z$bfr];
2605
     z$cm=1/2[z$bfl,z$tfr];
     z$ct=1/2[z$tfl,z$tr];
2606
      draw z$bl--z$br--z$tl--cycle;
2607
      draw z$bfl--z$bfr;
2608
      draw z$tfl--z$tfr;
2609
2610 \text{ enddef};
2611
```

hiderectangle Draws an invisible rectangular box of the usual dimensions that covers up anything underneath it.

```
2612 def hiderectangle(suffix $)(expr 1, h) =
2613 begingroup
2614 save c;
2615 pair c[];
2616 c1=(x$,y$);
     c2=c1+(1,0);
2617
     c3=c1+(1,h);
2618
     c4=c1+(0,h);
2619
2620
     unfilldraw c1--c2--c3--c4--cycle;
      endgroup
2621
2622 enddef;
2623
```

drawdashboxover Draws a dashed box that covers up anything underneath it.

```
2624 def drawdashboxover(suffix $)(expr l, h) =
2625 rectpoints($, l, h);
```

```
2626 hiderectangle($, l, h);
2627 z$c = 1/2[z$bl,z$tr];
2628 draw z$bl--z$br--z$tr--z$tl--cycle dashes;
2629 enddef;
2630
```

```
drawindexbox Draws an index box. The main box is 1 by h and the small box at the top left is 1p by hp. The main box points are the usual z$b1 etc, but the top box points are z$P.b1 etc. The str is put at the center (z$P.c) of the small box.
```

```
2631 def drawindexbox(suffix $)(expr 1, h, lp, hp)(text str) =
2632
     rectpoints($, 1, h);
2633
     z = 1/2[z$bl,z$tr];
     zP = ztl;
2634
     rectpoints($P, lp, hp);
2635
2636
      zP.c = 1/2[zP.bl, zP.tr];
2637
      draw z$bl--z$br--z$tr--z$tl--cycle;
2638
      draw z$P.bl--z$P.br--z$P.tr--z$P.tl--cycle;
2639
      label(str, z$P.c);
2640 \text{ enddef}:
2641
```

```
drawroundedbox Draws a rectangular box with rounded corners (like the LATEX \oval). The box is 1 by h and located by the bottom left corner. The corners are rounded with a radius of r. If the radius is too large for the box it is reduced so that at least two opposite sides are semi-circular. The str is put at the center (z$c) of the box.
```

```
2642 def drawroundedbox(suffix $)(expr l, h, r)(text str) =
2643 rectpoints($, l, h);
2644 begingroup
```

```
2645
      save rad;
      numeric rad; rad := r;
2646
      if rad > 1/2:
2647
        rad := 1/2;
2648
2649
     fi
      if rad > h/2:
2650
        rad := h/2;
2651
2652
      fi
      draw (x$br-rad, y$br){right}..{up}(x$br, y$br+rad)--
2653
2654
            (x$tr, y$tr-rad){up}..{left}(x$tr-rad, y$tr)--
2655
            (x$tl+rad, y$tl){left}..{down}(x$tl, y$tl-rad)--
            (x$bl, y$bl+rad){down}..{right}(x$bl+rad, y$bl)--cycle;
2656
2657
      endgroup;
      z$c = 1/2[z$bl,z$tr];
2658
      label(str, z$c);
2659
2660 enddef;
2661
```

```
drawovalbox Draws an elliptical box with horizontal diameter 1 and vertical diameter h. The box is located by the center point.
```

```
2662 def drawovalbox(suffix $)(expr 1, h)(text str) =
```

```
ellipsepoints($, 1, h);
2663
      z$c = 1/2[z$ml,z$mr];
2664
2665
      begingroup
      save p;
2666
2667
      path p;
2668
     p = fullcircle scaled h xscaled (1/h) shifted z$c;
2669
      draw p;
2670
     endgroup;
2671
     label(str, z$c);
2672 enddef;
2673
```

drawdashellipse Draws a dashed elliptical box with horizontal diameter 1 and vertical diameter h. The ellipse is located by the center point.

```
2674 def drawdashellipse(suffix $)(expr 1, h) =
      ellipsepoints($, 1, h);
2675
      z$c = 1/2[z$ml,z$mr];
2676
2677
      begingroup
2678
      save p;
2679
      path p;
     p = fullcircle scaled h xscaled (1/h) shifted z$c;
2680
2681
      draw p dashes;
2682
      endgroup
2683 enddef;
2684
```

drawdrum Draws a drum box with length 1 and height h. The top and bottom ellipse minor/major diamter ratio id drumlid. The text is put at the (curved) center of the drum.

```
2685 def drawdrum(suffix $)(expr 1, h)(text str) =
2686
      save vdia; numeric vdia;
      vdia := drumlid*l;
                                         % ellipse vertical diameter
2687
                                         % full & half ellipse paths
      save pf, ph; path pf, ph;
2688
2689
2690
      % points on the basic rectangle
      z = z;
2691
     z$tr = (x$+1, y$+h);
2692
2693
      z$br = (x$tr, y$);
     z$tl = (x$, y$tr);
2694
     z$ml = 1/2[z$bl,z$tl];
2695
2696
      z$mr = 1/2[z$br,z$tr];
2697
      z$tc = 1/2[z$tl,z$tr];
                                 % center of top rectangle line
2698
     z$bc = 1/2[z$bl,z$br];
2699
     % draw box sides
2700
     draw z$tl--z$bl; draw z$tr--z$br;
2701
     % points on top ellipse
2702
      z$T''' = z$tc;
                                 % ellipse center
2703
2704
      ellipsepoints($T''', 1, vdia);
```

```
2705
     z$tm = z$T'''.tm;
2706
     z$tml = z$T'''.tl;
     z$tmr = z$T'''.tr;
2707
2708
     % points on bottom ellipse
2709
     z$B''' = z$bc;
2710
     ellipsepoints($B''', 1, vdia);
2711
     z$bm = z$B''.bm;
2712
     z$bml = z$B'''.bl;
2713
     z$bmr = z$B'''.br;
2714
2715
2716
     % box center point
2717
     z$c = 1/2[z$T''.bm, z$B''.bm];
2718
2719
     % draw top ellipse
     pf = fullcircle scaled vdia xscaled (l/vdia) shifted z$T''';
2720
2721
     draw pf;
2722 % draw bottom half ellipse
2723
     ph = (halfcircle rotated 180) scaled vdia xscaled (l/vdia) shifted z$B''';
2724
     draw ph;
    label(str, z$c);
2725
2726 enddef;
2727
```

drawoutputbox Draws an output box. The box is 1 by h and the str is put at the center. The bottom of the box is a wavy line.

This code was supplied by Guy Worthington (see comp.text.tex newsgroup thread *Trial, ignore*, January 2004, and in particuar Guy's message of 2004/01/27).

```
2728 def drawoutputbox(suffix $)(expr 1, h)(text str) =
2729
     rectpoints($, 1, h);
     begingroup
2730
2731
        save c;
2732
        pair c[];
        c1 = (x$br, y$br+1/8h); % right side of box is shorter
2733
        c2 = (x$bm, y$bm+1/16h);
2734
        % draw the tear
2735
        draw z$bl..c2{dir 45}..c1{dir -15}--z$tr--z$tl--cycle;
2736
2737
      endgroup;
2738
      z = (x$bm, 1/2(y$bm+y$tm));
      label(str, z$c);
2739
2740 \text{ enddef};
2741
```

drawstickman Draws a full frontal genderless stick figure, inside a rectangle 1 by h.

```
2742 \def drawstickman(suffix $)(expr 1, h) =
2743 rectpoints($,1,h);
2744 begingroup
2745 save c;
```

```
pair c[];
2746
2747
        c1 = 8/24[z$bm,z$tm];
        c2 = 15/24[z$bm,z$tm];
2748
        c3 = 18/24[z$bm,z$tm];
2749
        c4 = 1/2[c3,z$tm];
2750
2751
        c6 = (x$bl, ypart(c2));
2752
        c7 = (x\$br, ypart(c2));
        draw z$bl--c1--z$br;
2753
                                              % legs
        draw c1--c3;
                                              % body
2754
        draw c6--c7;
                                              % arms
2755
        draw c3{right}..z$tm{left}..cycle; % head
2756
2757
      endgroup;
2758 enddef;
2759
```

The end of the package

2760 (/up)

References

- [EN89] R. Elmasri and S. B. Navathe. Fundamentals of Database Systems. Benjamin Cummings Publishing Co. Inc., 1989.
- [GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The La-TeX Companion*. Addison-Wesley Publishing Company, 1994.
- [GRM97] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The LaTeX Graphics Companion*. Addison-Wesley Publishing Company, 1997.
- [Hob92] John D. Hobby. A user's manual for MetaPost. Computing Science Technical Report no. 162, AT&T Bell Laboratories, Murray Hill, NJ, April 1992. (Available from CTAN with the MetaPost distribution in .../graphics/metapost).
- [Hoe98] Alan Hoenig. TeX Unbound. Oxford University Press, 1998.
- [IDE85] AFWAL/MLTC, Wright-Patterson AFB, OH. Integrated Informatiuon Support Systems (IISS), Vol. V: Common Data Model Subsystem, Part 4: Information Modeling Manual — IDEF1X. Report Number: AFWAL-TR-86-4006, Volume V, 1985.
- [ISO87] ISO TR9007. Information processing systems Concepts and terminology for the conceptual schema and the information base, 1987.
- [ISO94] ISO 10303-11:1994. Industrial automation systems and integration Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual, 1994.

- [Knu86] Donald E Knuth. *The METAFONTbook*. Addison-Wesley Publishing Company, 1986.
- [Lam94] Leslie Lamport. LATEX: A Document Preparation System. Second edition. Addison-Wesley Publishing Company, 1994.
- [NH89] G. M. Nijssen and T. A. Halpin. Conceptual Schema and Relational Database Design. Prentice Hall, 1989.
- [Rec97] Keith Reckdahl. Using EPS Graphics in LaTeX2e Documents. February 1997. (Available from CTAN as ../info/epslatex.ps).
- [RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen. Object-Oriented Modeling and Design Prentice Hall, 1991.
- [SW94] Douglas A. Schenck and Peter R. Wilson. *Information Modeling the EXPRESS Way.* Oxford University Press, 1994. (ISBN 0-19-508714-3)
- [SM88] S. Shlaer and S. J. Mellor. Object-Oriented System Analysis. Yourdon Press, 1988.
- [Wil99] Peter Wilson. Some Experiences in Running METAFONT and MetaPost. November, 1999. (Available from CTAN as .../info/metafp.ps).

Note: See http://www.tug.org for information on accessing CTAN — the Comprehensive T_{EX} Archive Network.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	С	D
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	1381, 1387, 1393, 1399, 1405, 1411, 1417, 1423, 1429 \centering 1367, 1373, 1379, 1385, 1391, 1397, 1403, 1409, 1415, 1421, 1427 \changes 2513, 2535	$\begin{tabular}{lllllllllllllllllllllllllllllllllll$
В	circpoints (r) <u>1727</u>	$\texttt{defaultsmoothrad}\;(v)$
\bf 1018, 1038, 1044, 1052	\columnsep 1351	$\dots \dots \dots \dots \dots \underline{1540}$

dotdiam (v) 4, <u>1519</u>	drawEXPRESSION (r)	dı
dotlabel (r) 26	$\dots \dots \dots 11, \underline{2193}$	
dotpen (v) 4, <u>1532</u>	drawF (r) 8, <u>1858</u>	dı
dots (r) 7, <u>1672</u>	drawGENERIC (r) 11, <u>2201</u>	dı
dotspen (v) 4, <u>1532</u>	drawGEVENT (r) . $13, 2344$	dı
dotsscale (v) <u>1529</u>	drawgrid (r) 7, <u>1695</u>	dı
drawA (r) 13, 1839	drawindexbox (r) $16, 2631$	dı
drawASDT (r) \dots 2181	drawINTEGER (r) $11, 2201$	dı
drawBINARY (r) . $10, \overline{2193}$	drawISR (r) $12, 2307$	dı
drawBOOLEAN (r) $11, \overline{2193}$	drawISRR (r) 12, <u>2320</u>	dı
drawCA (r) \ldots 8, $\overline{\underline{1820}}$	drawISU (r) \dots 12, <u>2279</u>	dı
drawcardbox (r) \ldots 14	drawISUR (r) $12, 2292$	
drawCD (r) 14, <u>2425</u>	drawLEVENT (r) . $12, 2335$	dı
drawcircleA (r) $14, \overline{2481}$	drawLOGICAL (r) $11, 2201$	
drawcirclebox (r) $13, \overline{\underline{2355}}$	drawnormal (r) \dots 9, <u>1970</u>	dı
drawCircledDot (r) 14	drawnormalCA (r) $9, 2044$	dı
drawCOMPLEX (r) $11, 2193$	drawnormalCD (r) $14, 2469$	
drawD (r) $\dots 8, \underline{1785}$	drawnormalD (r) . $9, 2013$	
drawdash (r) $8, \underline{1930}$	drawnormalDCA (r) . $\underline{2457}$	e]
drawdashA (r) $14, 2445$	drawnormalDD (r) $9, 2032$	er
drawdashboxover (r) .	drawnormalF (r) . $9, 2062$	١¢
	drawnormalFO (r) 9, <u>2068</u>	
drawdashcircle (r)	drawnormalfour (r) $9, \underline{1981}$	e٦
	drawnormalfourCA $({ m r})$	e٦
drawdashellipse (r) .	$\dots \dots \dots g, \underline{2056}$	
$\dots \dots \dots \dots \dots \dots 16, \underline{2674}$	drawnormalfourD $({ m r})$.	
drawdashfour (r) $8, \underline{1940}$	$\dots \dots \dots g, \underline{2025}$	t/
drawdashfourO (r) $9, \underline{1910}$	drawnormalfourO $({ m r})$.	t/
drawdash0 (r) $8, \frac{1945}{1945}$	$\dots \dots \dots g, \underline{1999}$	
drawdashOA (r) . 14, 2451	drawnormalO (r) . 9, <u>1986</u>	
drawdash00 (r) $ 9, \underline{1963}$	drawnormalOA (r) $9, 2038$	ga
drawdashthree (r) $8, \underline{1935}$	drawnormalOD (r) $14, 2463$	ga
drawdashthreeO (r) $8, \underline{1950}$	drawnormalOO (r) 9, 2006	go
drawDCA (r) \dots 13, 2383	drawnormalthree (r) .	go
drawdiamondbox (r) $15, 2505$	$\dots \dots \dots g, \underline{1975}$	gf
15, <u>2554</u>	drawnormalthreeCA (r)	gf
drawDot (r) 14, <u>2497</u>	$\dots \dots \dots g, \underline{2050}$	
drawdots (r) $8, \underline{1877}$	drawnormalthreeD (r)	
		\ł
drawdotsfour (r) 8, <u>1891</u> drawdotsfourO (r) 8, <u>1914</u>	drawnormalthreeO (r)	\ł
		hi
drawdots0 (r) 8, $\underline{1898}$	drawNUMBER (r) . 11, <u>2201</u>	Η٦
drawdots00 (r) 8, <u>1922</u>	draw0 (r) $8, \frac{1769}{1200}$	H3
drawdotsthree (r) $8, \frac{1884}{1000}$	drawOA (r) $8, \frac{1800}{2250}$	
drawdotsthreeO (r) 8, <u>1906</u>	drawOB (r) $\dots \dots 2256$	
drawdoublerectangle	drawOD (r) $13, 2404$	\i
(r) $15, 2591$	drawoutputbox (r) $16, \underline{2728}$	/i
drawdrum (r) 16, <u>2685</u>	drawovalbox (r) $16, 2662$	
drawENT (r) 12, <u>2247</u>	drawPREF (r) $12, 2272$	
drawENUM (r) 11, <u>2214</u>	drawREAL (r) 11, <u>2209</u>	
	())	

drawroundedbox (r) *16*, <u>2642</u> lrawSCHEMA (r) . 10, 2159lrawSDT (r) $\ldots \ldots 2169$ lrawSELECT (r) . 11, <u>2226</u> lrawstickman (r) 16, 2742lrawSTRING (r) . 11, 2209lrawthick (r) \dots 9, 2074lrawthickO (r) .. 9, <u>2081</u> lrawthickOO (r) . 9, <u>2089</u> lrawtriplerectangle (r) *15*, <u>2600</u> lrawtwodiamondbox (r) *15*, <u>2566</u> lrawTYPE (r) ... 12, <u>2238</u> lrumlid (v) 4, <u>1544</u> \mathbf{E} ellipsepoints (r) . $\underline{2369}$ enth (v) 5, 1594evensidemargin 1354, 1362 eventh (v) 5, <u>1611</u> eventslope (v) .. 5, <u>1611</u> \mathbf{F} footnotesep \dots 1358 footskip $\dots \dots \dots 1348$ \mathbf{G} gab (v) 4, <u>1554</u> gal (v) 4, <u>1554</u> gdb (v) 13, <u>2363</u> gdl (v) *13*, <u>2363</u> gfb (v) 4, <u>1554</u> gfl (v) 4, <u>1554</u> \mathbf{H} headheight $\dots 1345$ headsep $\dots \dots \dots \dots 1346$ niderectangle (r) $15, \underline{2612}$ IvH (r) 6, <u>1642</u> IxH (r) 7, <u>1656</u> Ι ifpdf 1329, 1336 includegraphics 1368, 1374, 1380,1386,1392, 1398,

	1404,	141	0,
	1416,	1422,	1428
ish(v)		5	, <u>1600</u>
isrh (v	7)	5	, <u>1600</u>
\it .	106	0, 1066	, 1071

\mathbf{L}

label (r) 26

\mathbf{M}

$marginparpush \dots 1357$
marginparsep 1356
$\mbox{marginparwidth}$ 1355
maxx (v) 4, <u>1514</u>
maxy (v) $4, \underline{1514}$
mediation (r) 20

\mathbf{N}

\n 1472, 1473
namespace (r) 7, <u>1663</u>
\nd 2507, 2530
$\texttt{ndextra}(v) \dots 5, \underline{1623}$
\newif $\ldots \ldots 1329$
nextra (v) 5, <u>1617</u>
niextra (v) 5, <u>1617</u>
normalpen (v) $4, \underline{1532}$
normalpensize (v) . <u>1524</u>
\nTry 1472, 1473

0
\oddsidemargin
$\dots \dots 1353, 1361$
onelineh (v) 5, <u>1562</u>

\mathbf{P}

pconh (v) $5, 1589$
pconl (v) $5, 1585$
\pdffalse 1331
\pdfoutput . 1330, 1337
\pdftrue 1333
pickup (r) 26
prologues (v) <i>61</i>

\mathbf{R}

rectpoints (r) \dots <u>1700</u>
rhompoints (r) \dots <u>1713</u>
rotatedaround (r) 24
routines:
++
\sim

circpoints 1727
dashedgrid 1676
dashes $7 \frac{1668}{1668}$
circpoints <u>1727</u> dashedgrid <u>1676</u> dashes 7, <u>1668</u> dotlabel <i>26</i>
dots 7, <u>1672</u>
drawA 13, 1839
drawASDT $\dots 2181$
drawASDT <u>2181</u> drawBINARY . <i>10</i> , <u>2193</u>
drawBOOLEAN $11, \overline{2193}$
drawCA $8, \frac{1820}{1820}$
drawcardbox 14
drawCD $1/2425$
drawCD 14, <u>2425</u>
drawcircleA 14, 2481
drawcirclebox <i>13</i> , <u>2355</u>
$\dots \dots \dots \dots \dots 13, \underline{2355}$
drawCircledDot . 14
drawCOMPLEX 11, <u>2193</u>
drawD 8, <u>1785</u>
$\frac{1100}{2}$
drawdash 8, <u>1930</u> drawdashA 14, <u>2445</u>
drawdashA 14, <u>2445</u>
drawdashboxover
$\dots \dots 15, \underline{2624}$
drawdashcircle .
<i>13</i> , <u>2475</u>
drawdashellipse
16 2674
16, 2674
<i>16</i> , <u>2674</u> drawdashfour <i>8</i> , 1940
16, <u>2674</u> drawdashfour 8, <u>1940</u> drawdashfour0 9, <u>1957</u>
16, <u>2674</u> drawdashfour 8, <u>1940</u> drawdashfour0 9, <u>1957</u> drawdash0 8, <u>1945</u>
$\begin{array}{cccccccccccccccccccccccccccccccccccc$
$\begin{array}{cccccccccccccccccccccccccccccccccccc$
$\begin{array}{cccccccccccccccccccccccccccccccccccc$
16, <u>2674</u> drawdashfour 8, <u>1940</u> drawdashfour0 9, <u>1957</u> drawdash0 8, <u>1945</u> drawdash0A . 14, <u>2451</u> drawdash00 9, <u>1963</u> drawdashthree 8, <u>1935</u> drawdashthree0 . 8, <u>1951</u> drawDCA 13, <u>2383</u>
16, <u>2674</u> drawdashfour 8, <u>1940</u> drawdashfour0 9, <u>1957</u> drawdash0 8, <u>1945</u> drawdash0A. 14, <u>2451</u> drawdash0O 9, <u>1963</u> drawdashthree 8, <u>1935</u> drawdashthree0 8, <u>1951</u> drawDCA 13, <u>2383</u> drawdiamondbox .
16, <u>2674</u> drawdashfour 8, <u>1940</u> drawdashfour0 9, <u>1957</u> drawdash0 8, <u>1945</u> drawdash0A. 14, <u>2451</u> drawdash0O 9, <u>1963</u> drawdashthree 8, <u>1935</u> drawdashthree0 8, <u>1951</u> drawDCA 13, <u>2383</u> drawdiamondbox .
16, <u>2674</u> drawdashfour 8, <u>1940</u> drawdashfour0 9, <u>1957</u> drawdash0 8, <u>1945</u> drawdash0A. 14, <u>2451</u> drawdash00 9, <u>1963</u> drawdashthree 8, <u>1935</u> drawdashthree0 8, <u>1951</u> drawDCA 13, <u>2383</u> drawdiamondbox 15, <u>2554</u> drawDot 14, <u>2497</u>
$\begin{array}{cccccccccccccccccccccccccccccccccccc$

drawENUM .. 11, 2214 drawEXPRESSION *11*, <u>2193</u> drawF 8, <u>1858</u> drawGENERIC 11, 2201 drawGEVENT . 13, <u>2344</u> drawgrid ... 7, 1695 drawindexbox 16, 2631 drawINTEGER 11, 2201 drawISR ... 12, <u>2307</u> drawISRR .. 12, <u>2320</u> drawISU ... 12, 2279 drawISUR .. 12, <u>2292</u> drawLEVENT . 12, <u>2335</u> drawLOGICAL 11, 2201 drawnormal . . 9, <u>1970</u> drawnormalCA 9, 2044drawnormalCD 14, 2469drawnormalD . 9, 2013 drawnormalDCA . 2457drawnormalDD 9, 2032 drawnormalF . 9, 2062drawnormalFO 9, 2068 drawnormalfour . drawnormalfourCA drawnormalfourD drawnormalfour0 drawnormalO . $9, \underline{1986}$ drawnormalOA 9, 2038 drawnormalOD 14, 2463drawnormal00 9, 2006 drawnormalthree drawnormalthreeCA drawnormalthreeD drawnormalthree0drawNUMBER . 11, 2201 draw0 8, <u>1769</u> drawOA 8, <u>1800</u> drawOB <u>2256</u> drawOD 13, <u>2404</u> drawoutputbox *16*, <u>2728</u>

drawovalbox $16, \underline{2662}$	sdtbges (v) $\ldots 5, \underline{1579}$	gab 4, 1554
drawPREF 12, <u>2272</u>	sdtbh (v) 5, <u>1566</u>	gal 4, <u>1554</u>
drawREAL 11, 2209	sdtbl (v) 5, <u>1566</u>	gdb 13, <u>2363</u>
drawroundedbox .	sdtbs (v) 5, <u>1566</u>	gdl 13, <u>2363</u>
	sdtebh (v)	gfb $4, 1554$
drawSCHEMA . 10, 2159	sdtebl (v) 5	gfl 4, <u>1554</u>
drawSDT 2169	sdtebs (v) 5	ish 5, <u>1600</u>
drawSELECT . 11, 2226		
,	sharply (r) $7, 1757$	isrh $5, \frac{1600}{1514}$
drawstickman 16, <u>2742</u>	shifted (r) 20	maxx $4, \frac{1514}{1514}$
drawSTRING . 11, 2209	smooth (r) $10, 2097$	maxy $4, \frac{1514}{1000}$
drawthick \dots 9, 2074	smoothdash (r) . $10, \underline{2119}$	ndextra 5, <u>1623</u>
drawthickO \dots 9, $\underline{2081}$	smoothdots (r) . $10, \underline{2137}$	nextra $5, 1617$
drawthickOO . $9, \underline{2089}$	smoothly (r) 7, <u>1763</u>	niextra $5, 1617$
drawtriplerectangle	smoothrad (v) \dots 4, <u>1540</u>	normalpen 4, $\underline{1532}$
<i>15</i> , <u>2600</u>	smoothtwo (r) 10, <u>2114</u>	normalpensize . $\underline{1524}$
drawtwodiamondbox		onelineh $5, 1562$
<i>15</i> , <u>2566</u>	Т	pconh \dots $5, \overline{1589}$
drawTYPE $12, \overline{2238}$	$textheight \dots 1350$	pconl 5, <u>1585</u>
ellipsepoints . $\overline{2369}$	\textwidth 1349	prologues 61
hiderectangle	thickpen (v) 4, <u>1532</u>	schemah 5, <u>1606</u>
	thinpen (v) 1532	sdtbeh <u>1573</u>
HvH $6, \underline{1642}$	thinpenpen (v) 4	sdtbel <u>1573</u>
HxH $7, \underline{1656}$	\topmargin . 1352, 1360	sdtbes 1573
label	\topskip 1347	sdtbgeh 5, <u>1579</u>
mediation 20	\twolines	sdtbgel 5, <u>1579</u> sdtbgel 5, <u>1579</u>
	(tworthes	Subbger 0, 1019
7 1663	5 6 470 765 1130	adthman 5 1570
namespace $7, \underline{1663}$	5, 6, 470, 765, 1139	sdtbges 5, <u>1579</u>
pickup 26	5, 6, 470, 765, 1139 typeh (v) $\dots 5, \underline{1594}$	sdtbh $5, 1566$
pickup 26 rectpoints <u>1700</u>	typeh (v) 5, <u>1594</u>	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u>
pickup $\dots 26$ rectpoints $\dots 1700$ rhompoints $\dots 1713$	typeh (v) 5, <u>1594</u> U	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5, <u>1566</u>
pickup $\dots 26$ rectpoints $\dots 1700$ rhompoints $\dots 1713$ rotatedaround $\dots 24$	typeh (v) 5, <u>1594</u>	sdtbh
pickup	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u>	sdtbh
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V	sdtbh
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20 smooth 10, 2097	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables:	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5, <u>1566</u> sdtebh 5 sdtebl 5 sdtebl 5 sdtebs 5 smoothrad 4, <u>1540</u>
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20 smooth 10, 2097 smoothdash 10, 2119	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u>	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5, <u>1566</u> sdtebh 5 sdtebl 5 sdtebl 5 sdtebs 5 smoothrad 4, <u>1540</u> thickpen 4, <u>1532</u>
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20 smooth 10, 2097 smoothdash 10, 2119 smoothdots 10, 2137	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u>	sdtbh
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, <u>1757</u> shifted 20 smooth 10, <u>2097</u> smoothdash 10, <u>2119</u> smoothdots 10, <u>2137</u> smoothly 7, <u>1763</u>	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultfont 61	sdtbh
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, <u>1757</u> shifted 20 smooth 10, <u>2097</u> smoothdash 10, <u>2119</u> smoothldts 10, <u>2137</u> smoothly 7, <u>1763</u> smoothtwo 10, <u>2114</u>	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultfont 61 defaultgab <u>1548</u>	sdtbh
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, <u>1757</u> shifted 20 smooth 10, <u>2097</u> smoothdash 10, <u>2119</u> smoothdots 10, <u>2137</u> smoothly 7, <u>1763</u> smoothtwo 10, <u>2114</u> VH 6, <u>1628</u>	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultfont 61 defaultgab <u>1548</u> defaultgal <u>1548</u>	sdtbh
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20 smooth 10, 2097 smoothdash 10, 2119 smoothly 7, 1763 smoothtwo 10, 2114 VH 6, 1628 VhV 6, 1634	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultfont 61 defaultgab <u>1548</u> defaultgal <u>1548</u> defaultgfb <u>1548</u>	sdtbh
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, <u>1757</u> shifted 20 smooth 10, <u>2097</u> smoothdash 10, <u>2119</u> smoothdots 10, <u>2137</u> smoothly 7, <u>1763</u> smoothtwo 10, <u>2114</u> VH 6, <u>1628</u>	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultfont 61 defaultgab <u>1548</u> defaultgal <u>1548</u> defaultgfb <u>1548</u> defaultgf1 <u>1548</u>	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5, <u>1566</u> sdtebh 5 sdtebl 5 sdtebl 5 sdtebs 5 smoothrad 4, <u>1540</u> thickpen 4, <u>1532</u> thinpen 4 typeh 5, <u>1594</u> u 4, <u>1511</u> whatever 51
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20 smooth 10, 2097 smoothdash 10, 2119 smoothly 7, 1763 smoothtwo 10, 2114 VH 6, 1628 VhV 6, 1634	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultfont <u>61</u> defaultgab <u>1548</u> defaultgfb <u>1548</u> defaultgfl <u>1548</u> defaultgfl <u>1548</u> defaultgfl <u>1548</u> defaultgfl <u>1548</u>	sdtbh
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20 smooth 10, 2097 smoothdash 10, 2119 smoothly 7, 1763 smoothly 10, 2114 VH 6, 1628 VhV 6, 1644 VgV 6, 1649	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultfont 61 defaultgab <u>1548</u> defaultgfb <u>1548</u> defaultgf1 <u>1548</u> defaultgf1 <u>1548</u> defaultgf1 <u>1548</u> defaultgf1 <u>1540</u>	$\begin{array}{cccccccccccccccccccccccccccccccccccc$
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20 smooth 10, 2097 smoothdash 10, 2119 smoothdots 10, 2137 smoothly 7, 1763 smoothtwo 10, 2114 VH 6, 1628 VhV 6, 1634 VgV 70 ypart 70	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultfont 61 defaultgab <u>1548</u> defaultgal <u>1548</u> defaultgfb <u>1548</u> defaultgfl <u>1548</u> defaultgfl <u>1548</u> defaultgfl <u>1540</u> dotdiam 4, <u>1519</u>	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5, <u>1566</u> sdtebh 5 sdtebl 5 sdtebl 5 sdtebs 5 smoothrad 4, <u>1540</u> thickpen 4, <u>1532</u> thinpen 4 typeh 5, <u>1594</u> u 4, <u>1511</u> whatever 51 \vbox 6
pickup	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultfont 61 defaultgab <u>1548</u> defaultgal <u>1548</u> defaultgfb <u>1548</u> defaultgfl <u>1548</u> defaultgfl <u>1548</u> defaultgfl <u>1549</u> dotdiam 4, <u>1519</u> dotpen 4, <u>1532</u>	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5, <u>1566</u> sdtebh 5 sdtebl 5 sdtebl 5 sdtebs 5 smoothrad 4, <u>1540</u> thickpen 4, <u>1532</u> thinpen 4 typeh 5, <u>1594</u> u 4, <u>1511</u> whatever 51 \vbox 6 VH (r) 6, <u>1628</u> VhV (r) 6, <u>1649</u>
pickup 26 rectpoints <u>1700</u> rhompoints <u>1713</u> rotatedaround 24 sharply 7, <u>1757</u> shifted 20 smooth 10, <u>2097</u> smoothdash . 10, <u>2119</u> smoothdots . 10, <u>2137</u> smoothly 7, <u>1763</u> smoothly 7, <u>1763</u> smoothly 6, <u>1628</u> VhV 6, <u>1634</u> VyV 6, <u>1649</u> xpart 70 ypart 70 S schemah (v) 5, <u>1606</u>	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultgab <u>1548</u> defaultgab <u>1548</u> defaultgfb <u>1548</u> defaultgfb <u>1548</u> defaultgfl <u>1540</u> dotdiam 4, <u>1532</u> dotspen 4, <u>1532</u>	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5, <u>1566</u> sdtebh 5 sdtebl 5 sdtebl 5 sdtebs 5 smoothrad 4, <u>1540</u> thickpen 4, <u>1532</u> thinpen 4 typeh 5, <u>1594</u> u 4, <u>1511</u> whatever 51 \vbox 6 VH (r) 6, <u>1628</u> VhV (r) 6, <u>1649</u> W
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20 smooth 10, 2097 smoothdash . 10, 2119 smoothdots . 10, 2137 smoothly 7, 1763 smoothly 7, 1763 smoothtwo 10, 2114 VH 6, 1628 VhV 6, 1649 xpart 70 ypart 70 S schemah (v) 5, 1606 sdtbeh (v) 1573	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultfont 61 defaultgab <u>1548</u> defaultgfb <u>1548</u> defaultgfb <u>1548</u> defaultgfl <u>1548</u> defaultgfl <u>1549</u> dotdiam 4, <u>1532</u> dotspen 4, <u>1532</u> dotsscale <u>1529</u>	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5, <u>1566</u> sdtebh 5 sdtebl 5 sdtebl 5 sdtebs 5 smoothrad 4, <u>1540</u> thickpen 4, <u>1532</u> thinpen 4 typeh 5, <u>1594</u> u 4, <u>1511</u> whatever 51 \vbox 6 VH (r) 6, <u>1628</u> VhV (r) 6, <u>1634</u> VyV (r) 6, <u>1649</u> W \w 1479
pickup	typeh (v) 5, <u>1594</u> U u (v) 4, <u>1511</u> V variables: defaultdotdiam <u>1519</u> defaultdrumlid <u>1544</u> defaultfont 61 defaultgab <u>1548</u> defaultgfb <u>1548</u> defaultgfb <u>1548</u> defaultgfl <u>1540</u> dotdiam 4, <u>1519</u> dotspen 4, <u>1532</u> dotsscale <u>1529</u> drumlid 4, <u>1544</u>	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5, <u>1566</u> sdtebh 5 sdtebl 5 sdtebl 5 sdtebs 5 smoothrad 4, <u>1540</u> thickpen 4, <u>1532</u> thinpen 4 typeh 5, <u>1594</u> u 4, <u>1511</u> whatever 51 \vbox 6 VH (r) 6, <u>1628</u> VhV (r) 6, <u>1649</u> W
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20 smooth 10, 2097 smoothdash . 10, 2119 smoothdots . 10, 2137 smoothly 7, 1763 smoothly 7, 1763 smoothly 6, 1634 VH 6, 1634 VyV 6, 1649 xpart 70 ypart 70 S schemah (v) 5, 1606 sdtbeh (v) 1573 sdtbes (v) 1573	<pre>typeh (v) 5, 1594 U u (v) 4, 1511 V variables: defaultdotdiam 1519 defaultdrumlid 1544 defaultfont 61 defaultgab 1548 defaultgfb 1548 defaultgfb 1548 defaultgfl 1548 defaultgfl 1540 dotdiam 4, 1519 dotspen 4, 1532 dotsscale 1529 drumlid 4, 1544 enth 5, 1594</pre>	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5 sdtebh 5 sdtebh 5 sdtebh 5 sdtebs 5 sdtebs 5 smoothrad 4, <u>1540</u> thickpen 4, <u>1532</u> thinpen 4 typeh 5, <u>1594</u> u 4, <u>1511</u> whatever 51 \vbox 6, <u>1628</u> VhV (r) 6, <u>1634</u> VyV (r) 6, <u>1649</u> W \w 1479 whatever (v) 51
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20 smooth 10, 2097 smoothdash . 10, 2119 smoothdots . 10, 2137 smoothly 7, 1763 smoothly 7, 1763 smoothly 7, 1763 smoothtwo 10, 2114 VH 6, 1628 VhV 6, 1649 xpart 70 ypart 70 S schemah (v) 5, 1606 sdtbeh (v) 1573 sdtbes (v) 1573 sdtbes (v) 5, 1579	<pre>typeh (v) 5, 1594 U u (v) 4, 1511 V variables: defaultdotdiam 1519 defaultdordiam 1519 defaultdrumlid 1544 defaultgab 1548 defaultgab 1548 defaultgfb 1548 defaultgfb 1548 defaultgfl 1548 defaultgfl 1540 dotdiam 4, 1519 dotpen 4, 1532 dotsscale 1529 drumlid 4, 1544 enth 5, 1594 eventh 5, 1611</pre>	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5, <u>1566</u> sdtebh 5 sdtebl 5 sdtebl 5 sdtebs 5 smoothrad 4, <u>1540</u> thickpen 4, <u>1532</u> thinpen 4, <u>1532</u> thinpen 4 typeh 5, <u>1594</u> u 4, <u>1511</u> whatever 51 \vbox 6, <u>1628</u> VhV (r) 6, <u>1634</u> VyV (r) 6, <u>1649</u> W \w 1479 whatever (v) 51
pickup 26 rectpoints 1700 rhompoints 1713 rotatedaround 24 sharply 7, 1757 shifted 20 smooth 10, 2097 smoothdash . 10, 2119 smoothdots . 10, 2137 smoothly 7, 1763 smoothly 7, 1763 smoothly 6, 1634 VH 6, 1634 VyV 6, 1649 xpart 70 ypart 70 S schemah (v) 5, 1606 sdtbeh (v) 1573 sdtbes (v) 1573	<pre>typeh (v) 5, 1594 U u (v) 4, 1511 V variables: defaultdotdiam 1519 defaultdrumlid 1544 defaultfont 61 defaultgab 1548 defaultgfb 1548 defaultgfb 1548 defaultgfl 1548 defaultgfl 1540 dotdiam 4, 1519 dotspen 4, 1532 dotsscale 1529 drumlid 4, 1544 enth 5, 1594</pre>	sdtbh 5, <u>1566</u> sdtbl 5, <u>1566</u> sdtbs 5 sdtebh 5 sdtebh 5 sdtebh 5 sdtebs 5 sdtebs 5 smoothrad 4, <u>1540</u> thickpen 4, <u>1532</u> thinpen 4 typeh 5, <u>1594</u> u 4, <u>1511</u> whatever 51 \vbox 6, <u>1628</u> VhV (r) 6, <u>1634</u> VyV (r) 6, <u>1649</u> W \w 1479 whatever (v) 51

Y ypart (r) 70