A Form Exchange Language

Shun-ichi NAKAGAWA, Tomokazu ARITA, and Takeo YAKU

Last modification: 28, August, 2003

1 はじめに

仕様書支援システム Hiform のデータ構造を作成するために早急に外部データの構造を作成しなくてはならない．現在，Hiform の各様式はグラフ文法で定式化されており，様式はグラフによって内部構造を定式化している．そのため，外部ファイルはグラフの構造を記述できるファイル形式でなくてはならない．

グラフのデータ構造を記述する研究においては，木の構造を記述する H-Code や様々な図形を記述する PostScript などがある．

本研究では，H-code2 を基にした表の描画に必要な情報を持つグラフを記述する記述言語を作成する．その記述言語を FXL(Form eXchange Language) とよび，拡張 BNF により定義する．

2 H-code2[1]

H-code2 は，宮寺等により, 流れ図言語 Hichart の外部コードとして作成された木構造記述コードである．Hichart が，木構造であるため，H-code2 は木特有の階層構造，親子関係をデータにもつ．H-code2 の定義は，BNF で定義されており，テキスト形式のファイルである．

さらに，PostScript との連携を考え，フォント指定やカラーマネージメントを細かく記述することができる．

3 Form eXchange Language

本研究では，グラフの構造と表の描画に必要な情報を記述するファイル形式を定義する．構造は，拡張 BNF によって定義をし，記述言語を Form eXchange Language (FXL) と呼ぶ．

FXL は H-code2 を参考にするが，いくつか異なる点がある．H-code2 は，木構造を記述するのに対し，FXL は一般的な（閉路や多重辺を含む）グラフ

を記述することを目的としている．つまり，階層構造や親子関係は存在せず，頂点と辺を基本とする．

また，グラフのデータ以外に，Hiform 仕様書に必要なデータを記述できるよう配慮する．

3.1 仕様

Hifrom の各仕様の外部フォーマットとして拡張できるようにする．
グラフ描画のための汎用フォーマットを作る．
処理系内部のデータ構造とは別である．
テキストファイルである．
ファイルの拡張子は，”fxl” とする．
現在 Version は，version1.0 である．

3.2 記号定義

0xa	: 16 進数 a
0a	: 8 進数 a
'a'	: 文字 a
"a"	: 文字列 a
a ～ b	: a から b までの範囲の文字（英字，数字のみに適用）
a - b	: 文字集合 a から文字集合（文字)b を除いたもの
<a>	: 文章 a で表す文字集合（文字，文字列）
[a]	: a は省略可能
(a)	: a（記号の優先順位確定用）
{a,b,...}	: 文字の集合
a | b	: a か b のどちらか
a*	: a の 0 回以上繰り返し
a+	: a の 1 回以上繰り返し

BL	≡ < 警告音を表す文字（ASUCII コードでは 0x07）>
BS	≡ < バックスペースを表す文字（ASUCII コードでは 0x08）>
FF	≡ < 用紙送りを表す文字（ASUCII コードでは 0x0c）>
LF	≡ < 復帰改行を表す文字（ASUCII コードでは 0x0a）>
CR	≡ < 復帰を表す文字（ASUCII コードでは 0x0d）>
HT	≡ < 水平タブを表す文字（ASUCII コードでは 0x09）>
VT	≡ < 垂直タブを表す文字（ASUCII コードでは 0x0b）>
SPC	≡ < 空白を表す文字（ASUCII コードでは 0x20）>
EOL	≡ < 行の終わりを表す文字（文字列）>

ALL	≡ < 全ての文字 >
CH	≡ <SPC を除く印字可能文字（ASUCII コードでは 0x21～0x7e）>

SIGN	≡ '+' | '-'
DIGIT	≡ { '0'～'9' }
NDIGIT	≡ { '1'～'9' }
ODIGIT	≡ { '0'～'7' }
HDIGIT	≡ { '0'～'9', 'A'～'F', 'a'～'f' }
EXP	≡ ('E' | 'e') [SIGN] DIGIT+
ALPHA	≡ { 'A'～'Z', 'a'～'z', '_' }
DALPHA	≡ { '0' | '9', 'A' | 'Z', 'a' | 'z', '_' }
EXCH	≡ { EOL, CH – { ODIGIT, 'X', 'x' } }

3.3 ファイル構成要素（字句）

各字句区切り

DELIM ≡ { FF, LF, CR, HT, VT, SPC, EOL }

コメント

COMMENT ≡ ”/*” < ”*/” を含まない ALL* > ”*/”
| ”//” (ALL - EOL)* EOL

DEC は 10 進数，OCT は 8 進数，HEX は 16 進数，FLOAT は実数を表す.

DEC ≡ [SIGN] ’0’ | [SIGN] NDIGIT DIGIT*
OCT ≡ [SIGN] ’0’ ODIGIT+
HEX ≡ [SIGN] ’0’ (’X’ | ’x’) HDIGIT
FLOAT ≡ [SIGN] (’,’ DIGIT+ | DIGIT+ ’.’ DIGIT*) [EXP]
≡ [SIGN] DIGIT + EXP

アルファベット，アンダーバー，数字からなる文字列

STRING ≡ ALPHA DALPHA*

SRTNG で表せない文字列

STRING2 ≡ ’” ’ ((ALL - { ’ \ ’, ’” ’ })
| (’\’ EXCH)
| (”\x” <2 桁の HDIGIT>)
| (’\’ <3 桁の ODIGT>)
)*
’” ’

ただし，STRING2 中の次の文字列（文字）は特別な意味をもつ.

” \ ” ” : ’” ’ を表す.
” \\ ” : ’\’ を表す.
” \a ” : BL を表す.
” \b ” : BS を表す.
” \f ” : FF を表す.
” \l ” : LF を表す.
” \r ” : CR を表す.
” \t ” : HT を表す.
” \v ” : HM を表す.
” \n ” : EOL を表す.
” \x ”<2 桁の HDIGIT(=??) > :文字コード 0x??の文字を表す.
’ \ ’ <3 桁の ODIGIT(=???) > :文字コード 0???の文字を表す.
’ \ ’ < その他の EXCH> :予約（エラーとなる.）

各ブロック，拡張データの先頭

BOB ≡ ’ { ’ | ALPHA DALPHA* ’ { ’

”header{”	:ファイルのヘッドの先頭を表す.
”date{”	:ファイル更新日の先頭を表す.
”time{”	:ファイル更新時間の先頭を表す.
”application{”	:ファイル作成アプリケーション情報の先頭を表す.
”graph{”	:グラフの先頭を表す.
”graphHeader{”	:グラフのヘッドの先頭を表す.
”nodeSet{”	:頂点集合の先頭を表す.
”nodeObject{”	:頂点オブジェクトの先頭を表す.
”node{”	:頂点の先頭を表す.
”nodeID{”	:頂点 ID を表す.
”nodeX{”	:頂点の X 座標を表す.
”nodeY{”	:頂点の Y 座標を表す.
”nodeLabel{”	:頂点ラベルの先頭を表す.
”labelString{”	:ラベルの文字を表す.
”attribute{”	:属性の先頭を表す.
”cellSize{”	:セルサイズの先頭を表す.
”cellWidth{”	:セルの幅を表す.
”cellHeight{”	:セルの高さを表す.
”cellLocation{”	:セルの位置の先頭を表す.
”cellX{”	:セルの X 座標を表す.
”cellY{”	:セルの Y 座標を表す.
”cellColor{”	:セルの色の先頭を表す.
”fontRGB{”	:RGB での色の指定を表す.
”fontCMYK{”	:CMYK での色の指定を表す.
”fontLab{”	:Lab での色の指定を表す.
”fontGrayScale{”	:グレースケールでの色の指定を表す.
”parent{”	:親を表す.
”children{”	:子供を表す.
”depth{”	:深さを表す.
”edgeObject{”	:辺オブジェクトの先頭を表す.
”edge{”	:辺の先頭を表す.
”edgeID{”	:辺 ID を表す.
”startNode{”	:開始頂点を表す.
”endNode{”	:終端頂点を表す.
”edgeShape{”	:辺の形状を表す.
”edgeLabel{”	:辺ラベルの先頭を表す.
”< 上記以外 >{”	:予約（エラーとなる）

その他の字句

’,’ ’;’ ’}’　:区切り文字

3.4　ファイル構造 (構文)

BNF で表記する.

file → header list
list → ϵ | graph list
header → ”header{”
　　”date{” DEC ’,’ DEC ’,’ DEC ’}’
　　”time{” DEC ’,’ DEC ’,’ DEC ’}’
　　”application{” text text ’}’
　’}’
integer → DEC | OCT | HEX
real → DEC | OCT | HEX | FLOAT
text → STRING | STRING2

graph → ”graph{” graph_header graph_inner ’}’
graph_header → ”graphHeader{” graph_header_inner ’}’
graph_header_inner → ϵ
　　| date time graph_header_inner
　　| graph_name graph_header_inner
date → ”date{” DEC ’,’ DEC ’,’ DEC ’}’
time → ”time{” DEC ’,’ DEC ’,’ DEC ’}’
graph_name → ”graphName{” text ’}’
1 graph_inner → node_set edge_set

node_set → ”nodeSet{” node_list ’}’
node_list → ϵ | node_object node_list
node_object → ”nodeObject{” node_object_inner ’}’
node_object_inner → node node_label attribute cell_color parent children depth

node → ”node{”
　　node_id
　　node_position
　’}’
node_id → ”nodeID{” integer ’}’
node_position → ϵ | node_x node_y
node_x → ”nodeX{” integer ’}’
node_y → ”nodeY{” integer ’}’

node_label → "nodeLabel{" node_label_inner '}'
node_label_inner → label_string
label_string → ϵ |
"labelString{" label_string_inner '}'
label_string_inner → text

attribute → "attribute{" attribute_inner '}'
attribute_inner → cell_size cell_location cell_color
cell_size → "cellSize{" cell_size_inner '}'
cell_size_inner → cell_width cell_height
cell_width → "cellWidth{" integer '}'
cell_height → "cellHeight{" integer '}'
cell_location → "cellLocation{" cell_location_inner '}'
cell_location_inner → cell_x cell_y
cell_x → "cellX{" integer '}'
cell_y → "cellY{" integer '}'

cell_color → "cellColor{" cell_color_inner '}'
cell_color_inner → font_rgb | font_cmyk | font_hsl | font_grayscale

font_rgb → "fontRGB{" font_rgb_inner '}'
font_rgb_inner → FLOAT ',' FLOAT ',' FLOAT | text
font_cmyk → "fontCMYK{" FLOAT ',' FLOAT ',' FLOAT '}'
font_hsl → "fontHSL{" font_hsl_inner '}'
font_hsl_inner → FLOAT ',' FLOAT ',' FLOAT | text
font_grayscale → "fontGrayScale{" FLOAT '}'

parent → ϵ | "parent{" integer '}'
children → ϵ | "children{" integer '}'
depth → ϵ | "depth{" integer '}'

edge_set → "edgeSet{" edge_list '}'
edge_list → ϵ | edge_object edge_list
edge_object → "edgeObject{" edge_object_inner '}'
edge_object_inner → edge | edge edge_label edge_color

edge → ”edge{”
edge_id
start_node
end_node
edge_shape
’}’

edge_id → ”edgeID{” integer ’}’

start_node → ”startNode{” integer ’}’

end_node → ”endNode{” integer ’}’

edge_shape → ϵ | ”edgeShape{” edge_shape_inner ’}’

edge_shape_inner → integer | text

edge_label → ”edgeLabel{” edge_label_inner ’}’

edge_label_inner → label_string

edge_color → ”edgeColor{” edge_color_inner ’}’

edge_color_inner → font_rgb | font_cmyk | font_hsl | font_grayscale

3.5 FXL の各部分の詳細

3.5.1 ヘッド部分の解説

ヘッド部分には，ファイル全体の情報が書かれる．情報には，ファイルの更新日付・時間，作成アプリケーション情報等である．

構造

header{
date{ *year*, *month*, *day* }
time{ *hour*, *minute*, *second* }
application{ *application_name*, *application_version* }
}

date{ ··· }
year ファイル更新年
month ファイル更新月
day ファイル更新日

time{ ··· }
hour ファイル更新時間
minute ファイル更新分
second ファイル更新秒

application{ ··· }
application_name　　ファイル作成アプリケーション名
application_version　アプリケーションのバージョン

3.5.2　グラフ部分の解説

構造

graph{
　graphHeader{ date{ ··· } time{ ··· } graphName{ *graph_name* } }
　nodeSet{ nodeObject{ ··· } nodeObject{ ··· } ··· }
　edgeSet{ edgeObject{ ··· } edgeObject{ ··· } ··· }
}

graphHeader{ ··· }
グラフの全体に関する情報を記述

graphName{ *graph_name* }
graph_name : グラフ名 (text)

nodeSet{ ··· }
ノードに関する記述をする．nodeObject{} に関しては，後節で解説する．

edgeSet{ ··· }
辺に関する記述をする．edgeObject{} に関しては，後節で解説する．

3.5.3　ノード部分の解説

構造

nodeObject{
　node{
　　nodeID{ *ID_Number* }
　　nodeX{ x }
　　nodeY{ y }
　}
　nodeLabel{
　　labelString{ *label_string* }
　}

```
    attribute{
        cellSize{
            cellWidth{ width }
            cellHeight{ height }
        }
        cellLocation{
            cellX{ x }
            cellY{ y }
        }
    }
    cellColor{
        fontRGB{ R, G, B }
    }
}
```

nodeObject{ ··· }
１つのノードとそれに付随する情報を記述

node{ ··· }
１つのノードの情報を記述

nodeID{ *ID_Number* }
ノードの識別情報を記述
ID_Number：識別番号を表す（integer）

nodeX{ x }
ノードの X 座標を記述
x：X 座標を表す（integer）

nodeY{ y }
ノードの Y 座標を記述
y：Y 座標を表す（integer）

nodeLabel{ ··· }
ノードに付随するラベルの情報を記述

labelString{ *label_string* }
ラベルとして使用する文字列を記述

label_string：ラベルの文字列を表す（ text ）

attribute{ ··· }
ノードに付随する属性の情報を記述

cellSize{ ··· }
セルのサイズの情報を記述

cellWidth{ *width* }
セルの幅を記述
x：セルの幅を表す（ integer ）

cellHeight{ *height* }
セルの高さを記述
x：セルの高さを表す（ integer ）

cellLocation{ ··· }
セルの描画位置の情報を記述

cellX{ x }
セルの X 座標を記述
x：X 座標を表す（ integer ）

cellY{ y }
セルの Y 座標を記述
y：Y 座標を表す（ integer ）

cellColor{ *color* }
セルの色の情報を記述
color：以下の 4 種類の形式でセルの色を表す

fontRGB{ R, G, B }
RGB での色の指定を記述
R：R（赤）の値を表す（ '0'～'255' ）
G：G（緑）の値を表す（ '0'～'255' ）
B：B（青）の値を表す（ '0'～'255' ）

fontCMYK{ *C*, *M*, *Y*, *K* }
CMYK での色の指定を記述
C : C（シアン）の値を表す ('0'～'100')
M : M（マゼンダ）の値を表す ('0'～'100')
Y : Y（イエロー）の値を表す ('0'～'100')
K : K（ブラック）の値を表す ('0'～'100')

fontHLS{ *H*, *L*, *S* }
HLS での色の指定を記述
H : H（色相）の値を表す ('0'～'100')
L : L（明度）の値を表す ('0'～'100')
S : S（彩度）の値を表す ('0'～'100')

fontGrayScale{ *grayscale* }
GrayScale での色の指定を記述
grayscale : の値を表す ('0'～'100')

3.5.4 エッジ部分の解説

構造

```
edgeObject{
    edge{
        edgeID{ { edge_ID } }
        startNode{ { start_node } }
        endNode{ { end_node } }
        edgeShape{ { shape } }
    }
    edgeLabel{
        labelString{ label_string }
    }
    edgeColor{
        fontRGB{ C, M, Y, K }
    }
}
```

edgeObject{ ··· }
１つのエッジとそれに付随する情報を記述する

edgeID{ { edge_ID } }
エッジの ID を記述
{ edge_ID }：エッジ ID を表す (integer)

startNode{ { start_node } }
開始頂点を記述
{ start_node }：開始頂点を表す (integer)

endNode{ { end_node } }
終端ノードを記述
{ end_node }：終端ノードを表す (integer)

endShape{ { shape } }
エッジの形状を記述
{ shape }：エッジの形状を表す (text)

edgeLabel{ ··· }
エッジに付随するラベルの情報を記述

labelString{ *label_string* }
ラベルとして使用する文字列を記述
label_string：ラベルの文字列を表す (text)

edgeColor{ *color* }
エッジの色の情報を記述
color：RGB, CMYK, HLS, GrayScale の 4 種類の形式でエッジの色を表す

3.5.5 木に関する特別情報

構造

```
nodeObject{
    node{ cdots }
    parent{ { parent_node } }
    children{ { child_nodes } }
    depth{ { depth } }
}
```

parent{ { parent_node } }
親を記述
{ parent_node }：親ノードを表す (integer)

children{ { children_node } }
子を記述
{ children_nodes }：子ノードを表す (integer)

depth{ { depth } }
深さを記述
{ depth }：深さを表す (integer)

3.6 色を表す文字列の定義

”RGB” と”HSL” は，数値以外に文字列による入力を可能としている．その文字列の対応表を以下に示す.

また，表の数値は，見易さのために次のように定義する．
$0 \leq R < 256$, $0 \leq G < 256$, $0 \leq B < 256$
$0 \leq H \leq 360$, $0 \leq L \leq 100$, $0 \leq S \leq 100$
FXL 内では，数値はすべて 0 以上 1 以下の実数に変換して扱うものとする．
例: R の任意の値 r は，FXL 内では $r/256$ となる．

Color Name	R	G	B	H(°)	S(%)	L(%)
black	0	0	0	0	0	0
navy	0	0	128	240	100	25
darkblue	0	0	139	240	100	27
mediumblue	0	0	205	240	100	40
blue	0	0	255	240	100	50
darkgreen	0	100	0	120	100	19
green	0	128	0	120	100	25
teal	0	128	128	180	100	25
darkcyan	0	139	139	180	100	27
deepskyblue	0	191	255	195	100	50
darkturquoise	0	206	209	180	100	40
mediumspringgreen	0	250	154	156	100	49
lime	0	255	0	120	100	50
springgreen	0	255	127	149	100	50
aqua	0	255	255	180	100	50

Color Name	R	G	B	H(°)	S(%)	L(%)
cyan	0	255	255	180	100	50
midnightblue	25	25	112	240	63	26
dodgerblue	30	144	255	209	100	55
lightseagreen	32	178	170	176	69	41
forestgreen	34	139	34	120	60	33
seagreen	46	139	87	146	50	36
darkslategray	47	79	79	180	25	24
limegreen	50	205	50	120	60	50
mediumseagreen	60	179	113	146	49	46
turquoise	64	224	208	174	72	56
royalblue	65	105	225	225	72	56
steelblue	70	130	180	207	44	49
darkslateblue	72	61	139	248	39	39
mediumturquoise	72	209	204	177	59	55
indigo	75	0	130	274	100	25
darkolivegreen	85	107	47	82	38	30
cadetblue	95	158	160	181	25	50
cornflowerblue	100	149	237	218	79	66
mediumaquamarine	102	205	170	159	50	60
dimgray	105	105	105	0	0	41
slateblue	106	90	205	248	53	57
olivedrab	107	142	35	79	60	34
slategray	112	128	144	210	12	50
lightslategray	119	136	153	210	14	53
mediumslateblue	123	104	238	248	79	67
lawngreen	124	252	0	90	100	49
chartreuse	127	255	0	90	100	50
aquamarine	127	255	212	159	100	74
maroon	128	0	0	0	100	25
purple	128	0	128	300	100	25
olive	128	128	0	60	100	25
gray	128	128	128	0	0	50
skyblue	135	206	235	197	71	72
lightskyblue	135	206	250	202	92	75
blueviolet	138	43	226	271	75	52

Color Name	R	G	B	H(°)	S(%)	L(%)
darkred	139	0	0	0	100	27
darkmagenta	139	0	139	300	100	27
saddlebrown	139	69	19	25	75	30
darkseagreen	143	188	143	120	25	64
lightgreen	144	238	144	120	73	74
mediumpurple	147	112	219	259	59	64
darkviolet	148	0	211	282	100	41
palegreen	152	251	152	120	92	79
darkorchid	153	50	204	280	60	49
yellowgreen	154	205	50	79	60	50
sienna	160	82	45	19	56	40
brown	165	42	42	0	59	40
darkgray	169	169	169	0	0	66
lightblue	173	216	230	194	53	79
greenyellow	173	255	47	83	100	59
paleturquoise	175	238	238	180	64	80
lightsteelblue	176	196	222	213	41	78
powderblue	176	224	230	186	51	79
firebrick	178	34	34	0	67	41
darkgoldenrod	184	134	11	42	88	38
mediumorchid	186	85	211	288	58	58
rosybrown	188	143	143	0	25	64
darkkhaki	189	183	107	55	38	58
silver	192	192	192	0	0	75
mediumvioletred	199	21	133	322	80	43
indianred	205	92	92	0	53	58
peru	205	133	63	29	58	52
chocolate	210	105	30	25	75	47
tan	210	180	140	34	43	68
lightgrey	211	211	211	0	0	82
thistle	216	191	216	300	24	79
orchid	218	112	214	302	58	64
goldenrod	218	165	32	42	74	49
palevioletred	219	112	147	340	59	64
crimson	220	20	60	348	83	47

Color Name	R	G	B	H(°)	S(%)	L(%)
gainsboro	220	220	220	0	0	86
plum	221	160	221	300	47	74
burlywood	222	184	135	33	56	70
lightcyan	224	255	255	180	100	93
lavender	230	230	250	240	66	94
darksalmon	233	150	122	15	71	69
violet	238	130	238	300	76	72
palegoldenrod	238	232	170	54	66	80
lightcoral	240	128	128	0	78	72
khaki	240	230	140	54	76	74
aliceblue	240	248	255	208	100	97
honeydew	240	255	240	120	100	97
azure	240	255	255	180	100	97
sandybrown	244	164	96	27	87	66
wheat	245	222	179	39	76	83
beige	245	245	220	60	55	91
whitesmoke	245	245	245	0	0	96
mintcream	245	255	250	150	100	98
ghostwhite	248	248	255	240	100	98
salmon	250	128	114	6	93	71
antiquewhite	250	235	215	34	77	91
linen	250	240	230	30	66	94
lightgoldenrodyellow	250	250	210	60	80	90
oldlace	253	245	230	39	85	94
red	255	0	0	0	100	50
fuchsia	255	0	255	300	100	50
magenta	255	0	255	300	100	50
deeppink	255	20	147	327	100	53
orangered	255	69	0	16	100	50
tomato	255	99	71	9	100	63
hotpink	255	105	180	330	100	70
coral	255	127	80	16	100	65
darkorange	255	140	0	32	100	50
lightsalmon	255	160	122	17	100	73
orange	255	165	0	38	100	50

Color Name	R	G	B	H(°)	S(%)	L(%)
lightpink	255	182	193	350	100	85
pink	255	192	203	349	100	87
gold	255	215	0	50	100	50
peachpuff	255	218	185	28	100	86
navajowhite	255	222	173	35	100	83
moccasin	255	228	181	38	100	85
bisque	255	228	196	32	100	88
mistyrose	255	228	225	6	100	94
blanchedalmond	255	235	205	36	100	90
papayawhip	255	239	213	37	100	91
lavenderblush	255	240	245	340	100	97
seashell	255	245	238	24	100	96
cornsilk	255	248	220	48	100	93
lemonchiffon	255	250	205	54	100	90
floralwhite	255	250	240	40	100	97
snow	255	250	250	0	100	99
yellow	255	255	0	60	100	50
lightyellow	255	255	224	60	100	93
ivory	255	255	240	60	100	97
white	255	255	255	0	0	100

3.7 FXL の記述例

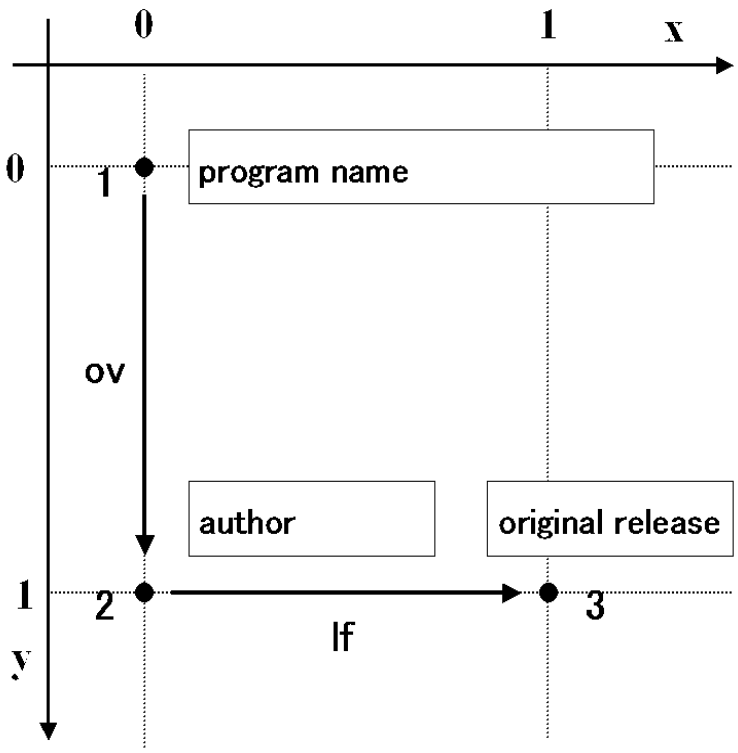

Fig. 1 Graph 1

```
\\ FXL version1.0

header{
    date{ 2000,11,24 }
    time{ 0,0,0 }
application{ "HiformED", "version 1.0" }
}

graph{
    graphHeader{
    date{ 2002,1,1 }
    time{ 0,0,0 }
    }

  nodeSet{
```

```
nodeObject{
  node{
    nodeID{ 1 }
    nodeX{ 0 }
    nodeY{ 0 }
  }
  label{
    labelString{“ program name ”}
  }
  attribute{
    cellSize{
      cellWidth{ 2 }
      cellHeight{ 1 }
    }
    cellLocation{
      cellX{ 0 }
      cellY{ 0 }
    }
  }
  cellColor{
    fontRGB{ 0, 0, 0 }
  }
}
nodeObject{
  node{
    nodeID{ 2 }
    nodeX{ 0 }
    nodeY{ 1 }
  }
  label{
    labelString{“ author ”}
  }
  attribute{
    cellSize{
      cellWidth{ 1 }
      cellHeight{ 1 }
    }
    cellLocation{
      cellX{ 0 }
```

```
        cellY{ 1 }
      }
    }
    cellColor{
      fontRGB{ 0, 0, 0 }
    }
  }
  nodeObject{
    node{
      nodeID{ 3 }
      nodeX{ 1 }
      nodeY{ 1 }
    }
    label{
      labelString{“ original release ”}
    }
    attribute{
      cellSize{
        cellWidth{ 1 }
        cellHeight{ 1 }
      }
      cellLocation{
        cellX{ 1 }
        cellY{ 1 }
      }
    }
    cellColor{
      fontRGB{ 0, 0, 0 }
    }
}

edgeSet{
  edgeObject{
    edge{
      edgeID{ 1 }
      startNode{ 1 }
      endNode{ 2 }
      edgeShapes\{“ arrow ”}
    }
```

```
      label{
        labelString{“ ov ”}
      }
      edgeColor{
        fontCMYK{ 0, 0, 0, 100 }
      }
    }
    edgeObject{
      edge{
        edgeID{ 2 }
        startNode{ 2 }
        endNode{ 3 }
        edgeShapes{ “ arrow ”}
      }
      label{
        labelString{“ lf ”}
      }
      edgeColor{
        fontCMYK{ 0, 0, 0, 100 }
      }
    }
  }
}
```

4　今後の課題

現在の FXL Version 1.0 では，基本的なグラフの構造を記述することが可能である．そして，Hifrom 仕様書の描画に必要な全ての情報を埋め込めむことも可能としている．

今後の課題は，さらに詳細な描画のための情報を記述できるようにすることを検討する必要がある．例えば，ラベルのフォントの指定，頂点・辺・ラベルの形状の情報等である．

また，Parser の作成をする必要がある．構文解析，内部コード生成部分等作成予定である．

参考文献

[1] Youzou Miyadera, Takeo Yaku, Hideaki konya, An Expression Method for Circulation od Tree–Structured Diagrams, 東京電機大学理工学部紀要 vol.19 No.1, 41-59 (1997).

[2] A.V. エイホ, R. セシィ, J.D. ウルマン共著, 原田賢一訳, コンパイラ原理・技法・ツール, サイエンス社 (1990).

[3] 小野木一樹, H コード言語解説書 ver. 5.0, Hichart 研究会資料 91-04 (1991).

[4] 有田友和, グラフに対するデータ構造記述フォーマット, プライベートメモ (2000).