グラフに対するデータ構造記述フォーマット

有田　友和 *

2000 年 5 月 29 日

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，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[3] や様々な図形を記述する PostScript などがある．

本研究では，H-code2[1] を基にした一般的なグラフを記述する記述言語を作成する．その記述言語を Graph Modeling Language とよび，拡張 BNF[2] 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　Graph Modeling Language

本研究では，グラフの構造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 Graph Modeling Language (GML) と呼ぶ．

*地球情報数理科学専攻，総合基礎科学研究科，日本大学

H-code2[1] は，木構造を記述するのに対し，GML 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，”gml” とする．
現在 Version は，0.01a 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.
”graphName{”	グラフの名前を表す.
”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.
”depth{”	深さを表す.
”parent{”	親を表す.
”children{”	子供を表す.
”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.
”label{”	ラベルの先頭を表す.
”labelString{”	ラベルの文字を表す.
”< 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_name
date → ”date{” DEC ’,’ DEC ’,’ DEC ’}’
time → ”time{” DEC ’,’ DEC ’,’ DEC ’}’
graph_name → ”graphName{” text ’}’
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 label parent children depth
node → ”node{”
node_id
node_position
’}’
node_position → ϵ | node_x node_y
node_id → ”nodeID{” integer ’}’
node_x → ”nodeX{” integer ’}’
node_y → ”nodeY{” integer ’}’
parent → ϵ | ”parent{” integer ’}’
children → ϵ | ”children{” integer ’}’
depth → ϵ | ”depth{” integer ’}’

label → ”label{” label_inner ’}’
label_inner → label_string
label_string → ϵ |
”labelString{” label_string_inner ’}’
label_string_inner → text
edge_set → ”edgeSet{” edge_list ’}’
edge_list → ϵ | edge_object edge_list
edge_object → ”edgeObject{” edge_object_inner ’}’
edge_object_inner → edge | edge labe
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

3.5 GML 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　ノード部分に解説

構造

nodeObject{
　node{
　　nodeID{ *ID_Number* }
　　nodeX{ x }

```
        nodeY{ y }
    }
    label{
        labelString{ label_string }
    }
}
```

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 ）

label{ ··· }
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 ）

3.5.4　エッジ部分に解説

構造

edgeObject{
    edge{
        edgeID{ { edge_ID } }
        startNode{ { start_node } }
        endNode{ { end_node } }
        edgeShape{ { shape } }
    }
    label{
        labelString{ *label_string* }
    }
}

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 } }
終端ノードを記述
{ shape }：終端ノードを表す (text)

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 GML の記述例

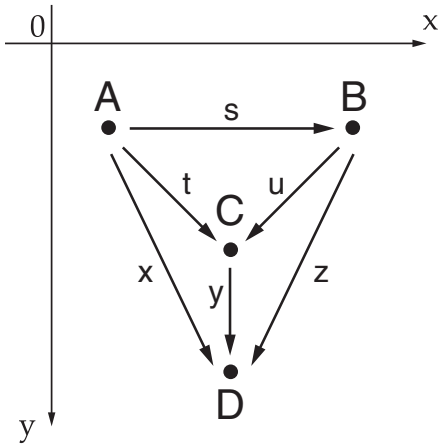

Fig. 1 Graph 1

```
\\ GML version0.01a

header{
    date{ 2000, 05, 08 }
    time{   14, 40, 00 }
    application{ "HiformED", "version 0.01a"}
}

graph{
    graphHeader{
        date{ 2000, 05, 08 }
        time{   14, 10, 00 }
    }

    nodeSet{
        nodeObject{
            node{
                nodeID{ 1 }
                nodeX{  0 }
                nodeY{  0 }
            }

            label{
                labelString{ "A" }
            }
        }

        nodeObject{
            node{
                nodeID{ 2 }
                nodeX{  2 }
```

```
            nodeY{  0 }
        }

        label{
            labelString{ "B" }
        }
    }

    nodeObject{
        node{
            nodeID{ 3 }
            nodeX{  1 }
            nodeY{  1 }
        }

        label{
            labelString{ "C" }
        }
    }

    nodeObject{
        node{
            nodeID{ 4 }
            nodeX{  1 }
            nodeY{  2 }
        }

        label{
            labelString{ "D" }
        }
    }

}

edgeSet{
    edgeObject{
        edge{
            edgeID{ 1 }
            startNode{ 1 }
            endNode{ 2 }
            edgeShape{ "arrow" }
        }

        label{
            labelString{ "s" }
        }
    }

    edgeObject{
        edge{
            edgeID{ 2 }
            startNode{ 1 }
            endNode{ 3 }
            edgeShape{ "arrow" }
```

```
    }

    label{
        labelString{ "t" }
    }
}

edgeObject{
    edge{
        edgeID{ 3 }
        startNode{ 2 }
        endNode{ 3 }
        edgeShape{ "arrow" }
    }

    label{
        labelString{ "u" }
    }
}

edgeObject{
    edge{
        edgeID{ 4 }
        startNode{ 1 }
        endNode{ 4 }
        edgeShape{ "arrow" }
    }

    label{
        labelString{ "x" }
    }
}

edgeObject{
    edge{
        edgeID{ 5 }
        startNode{ 3 }
        endNode{ 4 }
        edgeShape{ "arrow" }
    }

    label{
        labelString{ "y" }
    }
}

edgeObject{
    edge{
        edgeID{ 6 }
        startNode{ 2 }
        endNode{ 4 }
        edgeShape{ "arrow" }
    }
```

```
            label{
                labelString{ "z" }
            }
        }

    }

}
```

4 今後の課題

現在の GML Version 0.01a 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．

今後の課題は，まず Hifrom 仕様書の描画に必要な全ての情報を埋め込めるようにすること必要である．次に，詳細な描画のための情報を記述できるようにすることも検討必要がある．例えば，ラベルのフォントの指定，各頂点や辺の色の指定，ラベルの描画位置，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．今後は，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)