グラフ構造データのPMMLベース標準表現規格提案と実装
Proposal and Implementation of PMML Based Standard Representation on Graph Structured Data

光永悠紀*，鷲尾　隆，藤本　敦，元田　浩
Yuki Mitsunaga，Takashi Washio，Atsushi Fujimoto，Hiroshi Motoda
大阪大学産業科学研究所
I.S.I.R.，Osaka University

Abstract: The recent advanced techniques of data mininig analyse highly structured data such as sequences, trees and graphs, and also provide highly structured models and patterns in same manners. Under this trend, standard representations of the structure needs to be provided to ease their representation, their preprocessing and their exchange among data mining tools. The standard will further enforce the development of unified environments for data mining and new techniques for unseeked data structures. In this work, a standard representation of graph structures is proposed. Important factors to introduce the standard are the high extendability and expressibility of the structues and their associated features. Along this line, PMML based representation which is a subset of XML is designed in this work.

1　はじめに

従来の主要なデータマイニング技術は，各データが属性とクラスの値の組合せからなる表形式データやアイテムと呼ばれる記号の集合であるトランザクションデータをマイニング対象とするものであった．これに対し，Srikant や Agrawal はアイテムに階層的な種類分け (taxonomy) を導入しデータ中に多頻度に現れるアイテム種類集合を導出する手法 [1] や，アイテム間に順序関係がある場合の多頻度アイテム集合を導出する手法を開発した [2]．また Mannila 等は，トランザクションの系列からエピソードと呼ばれる多頻度に表れるアイテム集合系列を導出する手法の開発を行った [3]．更に新谷と喜連川等は，トランザクションを連続したアイテム系列と捉え，その中から多頻度に現れる連続アイテム系列を並列処理によって高速導出する手法を開発している [4]．トランザクションに限定せずに，一般の記号系列データから有限幅窓内に多頻度ないしは特徴的に現れる非連続性を含む系列パターンをマイニングする手法としては，BONSAI がある [5]．一方で近年，更に複雑な構造である木構造データの集合から多頻度な部分的構造をマイニングする手法の開発も盛んになっている．代表的なものとしては，順序木に関して多頻度パターンをマイニングする TreeMiner [6] や FREQT [7]，同様に非順序木に関してマイニングする手法 [8]，更にはルートノードが存在しない木 (Unrooted Tree) に関してマイニングする手法 [9] などが提案されている．

以上のような幾何学的構造として最も複雑なものの１つにグラフ構造があり，近年，グラフ構造など複雑な構造を持つデータに関するマイニング研究が，精力的にが進められつつある．Cook と Holder は Subdue [10]，吉田と元田は GBI [11] というグラフ構造マイニング手法を提案した．これらは頻度に限らず，種々の指標に基づいて特徴的な部分グラフ構造をマイニング可能でありかつ高速動作するが，greedy 探索を行うため完全探索はできない．これに対し，Dehaspe と Toivonen は ILP をベースとし，完全探索が可能な WARMR アルゴリズムを提案した [12]．更に Nijssen と Kok はグラフパターンの照合基準を緩めることで，より高速なアルゴリズム FARMER を提案した [13]．ILP ベースのアルゴリズムはグラフの vertex や edge に変数を導入できるなど，論理プログラミング上の大きなメリットを享受できる反面，探索範囲が膨大になるため比較的大きな部分構造のマイニングが難しい面があった．本来，大きなグラフや多数のグラフから共通する部分グラフ構造を発見する部分グラフ同型問題は，NP-完全の計算複雑性を持つことが知られており，大きな部分グラフ構造を探索すると照合の組合せ爆発に直面してしまう．このような理論的限界がありながらも，猪口，鷲尾，元田等はバスケット分析に用いられる Apripori アルゴリズムにグラフの隣接行列表現と代数演算を導入し，実用レベルの速度で多頻度部分グラフ構造を完全探索可

*連絡先：大阪大学産業科学研究所
〒 567-0047 大阪府茨木市美穂ヶ丘 8-1
E-Mail: mitsunaga@sanken.osaka-u.ac.jp

能な AGM アルゴリズムを提案した [14, 15]．これに引き続き，マイニングの高速化や扱える部分グラフ構造クラスの拡張に向けて，幾つもの研究がなされている [16, 17]．また現状はグラフに含まれる枝分かれの無い path の探索に限られるが，De Raedt と Kramer により頻度の多さや少なさというような anti-monotonic や monotonic な指標の組合せに基づいて，version space 上でグラフ内の部分構造を完全探索する MolFea というアルゴリズムが提案されている [18]．更には最近になって，グラフ構造・木構造・系列構造のマイニングに関するワークショップ (MGTS) [19] やより論理的な関係構造のマイニングに関するワークショップ (MRDM) [20] などが開催され，この分野の研究が盛んになってきている．

以上のようにデータマイニングで取り扱うデータ構造の複雑化に伴い，それらを柔軟で拡張が容易な形式により計算機内に格納・蓄積する方法に関する研究が開始されつつある．データマイニング研究では，効率的なアルゴリズムやマイニング指標にのみ注意が向けられがちであるが，データ表現に関する研究は，データマイニングシステムの実装のみならず，対象としうるデータ構造範囲の拡大や表現の体系化によって，多くの手法が共通して取り扱える基盤を提供する点で非常に重要である．現在，一般的な系列や木，グラフといったデータ構造について，汎用的な表現として述語論理をベースとした Progol が知られている [21]．しかしながら，論理プログラミングを中心としたアルゴリズム向けの表現であり，個々の構造データに新たな種類の属性や構造を柔軟に追加したり，データファイル全体の属性や必要な記載事項などを計算機が直接利用可能な形で記述することは難しい．また，表形式やトランザクションのような他のデータ構造と統一された表現原理に根ざした表現設計がなされていないため，実装上も多くのデータ構造を扱いかつそれら相互間のデータ変換や交換を行わねばならない，汎用のデータマイニングツールで取り扱う表現としては特殊な parser を用意せねばならず不便である．

これに対し，汎用データマイニングツールの製作者の立場から，様々なデータ形式やマイニング結果を統一的な枠組みの中で表現する標準規格を作る動きが生まれてきた．特に近年は新しいタグを追加的に定義して柔軟にデータ表現を拡張可能な XML データベースの研究が盛んである [22]．更にそれをデータマイニングの対象データ表現向けに設計した XML Table [23] やマイニング結果の XML 表現の世界標準規格 PMML を作ろうとする動きがある [24, 25]．特に後者は世界の主要な IT 企業が企画作りに参加しており，データマイニングの全体スキームやデータの流れの記述，前処理変換の記述に始まり，統計解析過程・結果の表現，Taxonomy の指定記述，マイニング結果として得られる Association Rules, Decision Trees, Center-Based and Distribution-Based Clustering, Regression, General Regression, Neural Networks, Naive Bayes, Sequences の XML による柔軟なデファクトスタンダード表現を提案している．PMML は完成された規格ではなく，データマイニング技術や XML 技術の発展に沿って，Verion 1.0 から始まり現在 Version 2.1 がリースされ，更に Version 3.0 の作成が進められている．なお，これらの規格の策定過程は SOURCEFORGE.NET というサイトの Public Forum でオープンにされており，第三者が問い合わせや意見を述べることができる．PMML は XML のサブセットドキュメントであり，各タグを定義するスキーマ部と実際のデータ部からなる．

しかしながら，先に述べたように最先端のデータマイニング手法は，系列や決定木などに留まらず，より一般的な木やグラフ構造などをマイニングの対象データとして扱い，かつまたマイニング結果としても出力するようになって来ている．このような状況下において，上述したデータ表現を一層拡張しこれら複雑なデータ構造の柔軟な標準表現を確立することが望まれる．これによって

1. データマイニングツールがマイニング対象とするデータの共有化
2. データマイニングツール同志のマイニング出力結果の交換・共有化
3. データマイニングに必要な前処理の容易化・効率化
4. データマイニング環境の統合化
5. データ表現の拡張性に対応する新たなマイニング手法・アルゴリズム開発促進

が期待される．本論では，上記の高度な構造データマイニング手法に対応したグラフ構造データの標準表現規格を提案する．この際，データ表現の柔軟な拡張や変更が可能で，かつデータ内容そのものを計算機に理解させて処理を実行させることが可能な XML 上に構築されたデータマイニング用のデファクトスタンダード表現である PMML に準拠し，更にそれによってグラフ構造を表現可能な規格を考えることとする．

2 PMML の概要

米国を中心とする世界的な IT 企業のデータマイニングツール開発部門を母体として，Data Mining Group というコンソーシアムが構成され，その中でデファクトスタンダードとしてデータマイニングに必要なデータの XML ベース表現規格が PMML として提案さてている．PMML は完成された規格ではなく，データマイニング技術や XML 技術の発展に沿って，Verion 1.0 から始まり現在 Version 2.1 がリリースされ，更に Version 3.0 の作成が進められている．なお，これらの規格の策定過程は SOURCEFORGE.NET というサイトの Public

Forum でオープンにされており，第三者が問い合わせや意見を述べることができる．

2.1 PMML

PMML は XML のサブセットドキュメントであり，各タグを定義するスキーマ部と実際のデータ部からなる．PMML は XML スキーマとデータ本体によって記述される．XML スキーマとはそのドキュメント自身が何であるかが，その内部に書かれているか或いはそれが記述された外部ファイル名が書かれている形式である．通常，これらの記述はドキュメントの先頭部分に置かれる．PMML はデータマイニングのためのデータやそのマイニング結果として得られたモデルを記述するドキュメントであるため，スキーマとしてはマイニングに使うデータ名やマイニング手法，それによって得られたモデル名などが記述される．

```
<xs:element name='PMML'>
 <xs:complexType>
  <xs:sequence>
   <xs:element ref='Header'/>
   <xs:element ref='MiningBuildTask'
    minOccurs='0' maxOccurs='1'/>
   <xs:element ref='DataDictionary'/>
   <xs:element ref='TransformationDictionary'
    minOccurs='0' maxOccurs='1'/>
   <xs:sequence minOccurs='0'
    maxOccurs='unbounded'>
    <xs:choice>
     <xs:element ref='TreeModel'/>
     <xs:element ref='NeuralNetwork'/>
     <xs:element ref='ClusteringModel'/>
     <xs:element ref='RegressionModel'/>
     <xs:element ref='GeneralRegressionModel'/>
     <xs:element ref='NaiveBayesModel'/>
     <xs:element ref='AssociationModel'/>
     <xs:element ref='SequenceModel'/>
    </xs:choice>
   </xs:sequence>
   <xs:element ref='Extension' minOccurs='0'
    maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='version' type='xs:string'
   use='required'/>
 </xs:complexType>
</xs:element>
```

上記は PMML の XML スキーマである．これはマイニング結果モデルの種類を表す．Version 2.1 ではここの書かれた８種類のモデルを記述できる．１つのドキュメントの中には１種類のモデルのみが記述されていてよいし，複数種類のモデルが記述されていてもよい．この部分において，ドキュメントが含むモデルの種類を指定する．$PMML$ タグ内にはこのドキュメントが表すモデルの著作権やどのようなデータからどのようなマイニング手法でどのような応用を目的として，またどのような改変履歴を持つか，各改変の日付など，このモデルの由来を記述した $Header$ タグがある．その次の $MiningBuildTask$ タグには，このモデルを生成したトレーニングのパラメータ設定などの条件が記載される．この定義は別スキーマとして記述される．また,$DataDictionary$ にはモデルを生成するの使用したデータの各属性フィールドやクラスなどの情報が書かれている．$TransformationDictionary$ には，元データからモデルを作る際にどのような前処理を施したかが記述される．例えば数値に０～１への規格化を施したとか，離散化を行った，値の置き換えを行った，値をグループ化したり平均値を取ったなどの前処理の詳細である．その定義が別スキーマとして記述される点は他と同様である．$< xs : choise >$ タグにはさまれた領域には各モデルを記述するスキーマ名を指定することができる．$Extension$ は，記述者が任意に拡張記述でる部分であり，通常はこのモデル構築を行った作業者名または会社名などや，モデルを特定のソフトウエアツール表示させる際の使用パラメータの指定など，様々な用途に使用可能なオプションである．また,$< xs : attribute >$ タグではこのドキュメントの属性として，使われている PMML バージョンやそのバージョン記述の型名が指定される．なお，$Header$ や $MiningBuildTask$，$DataDictionary$，$TransformationDictionary$ の記述内容に関する説明は割愛する．

2.2 マイニング結果モデル表現スキーマ

上記 PMML 記述内の各マイニング結果モデルを記述するスキーマ形式の定義は以下の通りである．

```
<xs:element name="ExampleModel">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Extension"
    minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="MiningSchema"/>
   <xs:element ref="ModelStats"
    minOccurs="0" maxOccurs="1"/>
   ...
   <xs:element ref="Extension"
    minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="modelName"
  type="xs:string" use="optional"/>
  <xs:attribute name="functionName"
   type="MINING-FUNCTION" use="required"/>
  <xs:attribute name="algorithmName"
   type="xs:string" use="optional"/>
 </xs:complexType>
</xs:element>
```

$ExampleModel$ のスキーマ内には，拡張オプション記述以外にどのような方法でマイニングを行ったかの $MiningSchema$ が記述される．これも別スキーマと

して定義される．具体的には元データのどの属性を用い，どの属性を用いなかったか，クラスを必要とするモデルの場合にはそのクラス属性名，外れ値や欠値の取り扱い方法などが記述される．$ModelStats$ には，そのモデルを構築するのに使用したデータ数やモデルの平均予測誤差などの統計的情報が記述される．$ExampleModel$ タグの属性 $< xs : attribute >$ にはモデル名 $modelName$，その機能名称 $functionName$，アルゴリズム名称 $algorithmName$ が記述される．属性 $functionName$ には $associationRules$, $sequences$, $classification$, $regression$, $clustering$ の５種類の機能を記述でき，このモデルが如何なる目的に使用可能な機能を有するかを示す．なお，$MiningSchema$ や $ModelStats$ の記述方法については割愛する..

2.3 データタイプスキーマ

上記ドキュメント記述内に現れるデータには，幾つかの基本タイプが存在する．それらは一般数値タイプ，整数値タイプ，高精度実数値タイプ，確率値タイプ，百分率タイプ，具体的な属性名を指定して値を定義するタイプなどである．例えば高精度実数値タイプのスキーマならば以下のように定義される．

```
<xs:simpleType name="REAL-NUMBER">
 <xs:restriction base="xs:double">
 </xs:restriction>
</xs:simpleType>
```

また以下のように配列を定義することも可能である．

```
 <Array n="3" type="int"> 1 22 3
 </Array>
 <Array n="3" type="string">
 ab  "a b"   "with \"quotes\" "
 </Array>
```

実データでは，n で配列の長さを指定し，type でそれが整数値配列か，実数値配列か，文字列の配列かを指定し，このタグの後中に実データを記述する．

2.4 アソシエーションルール記述の例

以下に PMML によるアソシエーションルール記述の例を示す．最初の３行は先に説明した通りである．次の $< DataDictionary >$ 内でデータ記述が２つあることを示し，このタグに挟まれた２行で transaction と item というデータが記号データであることを記述している．更にその次の $< AssociationModel >$ タグ内で機能がアソシエーションルールであり，４つのトランザクションデータから作られたモデルであり，アイテムは全部で３種類あり，最小支持度，最小確信度がそれぞれ 0.6，0.5，得られた多頻度集合が３つあり，得られたアソシエーションルールが２つあることを記述している．次の $< MiningSchema >$ タグに挟まれた部分では，$transaction$ と $item$ という１つの属性を持つデータからマイニングされたものであることが示されている．$<!--...-->$ はコメント行であり，その後，各アイテム名が記述されている．$< Itemset >$ タグ部分では $id =$ "1" のアイテムセットが支持度が１で，含まれるアイテム数が１であることなどが記述されている．更に $< AssociationRule >$ タグでは $id =$ "1" のアイテムセットがボディー $(antecedent)$，$id =$ "2" のアイテムセットがヘッド $(consequent)$ であるアソシエーションルールの支持度，確信度共に 1.0 であることなどが記されている．$< DataField >$，$< Item >$，$< Itemset >$，$< AssociationRule >$ などのタグのスキーマは，別途与えられるが割愛する．

```
<?xml version="1.0" ?>
<PMML version="2.1" >
<Header copyright="www.dmg.org" description=
 "example model for association rules"/>
<DataDictionary numberOfFields="2" >
<DataField name="transaction"
 optype="categorical" />
<DataField name="item" optype="categorical" />
</DataDictionary>
<AssociationModel
  functionName="associationRules"
  numberOfTransactions="4" numberOfItems="3"
  minimumSupport="0.6" minimumConfidence="0.5"
  numberOfItemsets="3" numberOfRules="2">
  <MiningSchema>
    <MiningField name="transaction"/>
    <MiningField name="item"/>
  </MiningSchema>
<!-- We have three items in our input data -->
<Item id="1" value="Cracker" />
<Item id="2" value="Coke" />
<Item id="3" value="Water" />
<!-- and two frequent itemsets
  with a single item -->
<Itemset id="1" support="1.0" numberOfItems="1">
  <ItemRef itemRef="1" />
</Itemset>
<Itemset id="2" support="1.0" numberOfItems="1">
  <ItemRef itemRef="3" />
</Itemset>
<!-- and one frequent itemset with two items. -->
<Itemset id="3" support="1.0" numberOfItems="2">
  <ItemRef itemRef="1" />
  <ItemRef itemRef="3" />
</Itemset>
<!-- Two rules satisfy the requirements -->
<AssociationRule support="1.0" confidence="1.0"
  antecedent="1" consequent="2" />
<AssociationRule support="1.0" confidence="1.0"
  antecedent="2" consequent="1" />
</AssociationModel>
</PMML>
```

3 グラフ構造表現のPMML提案

前記目的により，グラフ構造を表現するPMML規格の検討を行った．そのために，ますグラフ構造を定義するためのタグのスキーマを設計した．

3.1 スキーマ

スキーマを各タグについて個別に説明する．以下は，グラフモデルスキーマの定義である．最初の2行はXMLのバージョンとこのファイルの文字コード，XMLスキーマの説明を載せているホームページのURLを記述している．次の $< xs : element >$ タグに挟まれた部分では，グラフモデルが $< xs : complexType >$ という複数の内部タグから成り立つことを示している．その内部は $MiningSchema$，$ModelStats$，$Graph$ のスキーマを順番に参照する $< xs : sequence >$ タグと，モデル名を文字列かつオプションとして付与できること，機能としてマイニング機能を持つこと，アルゴリズム名を文字列かつオプションとして付与できることを定義する $< xs : attribute >$ タグから成っている．$< xs : sequence >$ タグ内では更に $< xs : element >$ タグが用いられ，$ModelStats$ スキーマの指定は最小無くてもよく最大で1つしか許されないことを示している．またシーケンスの最初と最後には，任意の個数の拡張参照スキーマを使用者が場合に応じて記述できるように工夫されている．

```
<?xml version="1.0" encoding="EUC-JP" ?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.dmg.org/PMML-2_1"
xmlns="http://www.dmg.org/PMML-2_1"
elementFormDefault="unqualified">
<xs:element name="GraphModel">
  <xs:complexType>
  <xs:sequence>
  <xs:element ref="Extension" minOccurs="0"
   maxOccurs="unbounded"/>
  <xs:element ref="MiningSchema"/>
  <xs:element ref="ModelStats" minOccurs="0"
   maxOccurs="1"/>
  <xs:element ref="Graph"/>
  <xs:element ref="Extension" minOccurs="0"
   maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="modelName"
   type="xs:string" use="optional"/>
  <xs:attribute name="functionName"
   type="MINING-FUNCTION"
   use="required"/>
  <xs:attribute name="algorithmName"
   type="xs:string" use="optional"/>
  </xs:complexType>
</xs:element>
```

上記で参照されているGraphスキーマの定義は以下の通りである．これも $< xs : sequence >$ タグと $< xs : attribute >$ タグから構成される $< xs : complexType >$ タグで記述されている．$< xs : sequence >$ タグ内では更に $< xs : element >$ タグを用いてグラフの $Virtex$ と $Edge$ のスキーマが参照されている．また，ここでもExtensionとして使用者が任意のスキーマを記述可能である．グラフ内には任意個のの $Virtex$ や $Edge$ が含まれるので $maxOccurs$ は上限なし $(unbounded)$ となっている．$< xs : attribute >$ タグでは各グラフデータを識別するための $graphId$ 名，グラフの種類とグラフマイニングによって得られるモデルの種類を示すための $graphType$，データ内のグラフ数を表す $recordCount$ が定義されている．いずれも対象がグラフデータなのか，グラフマイニングによって得られた結果としてのモデルなのかによって必要性が異なるので，オプションとなっている．

```
<xs:element name="Graph">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Extension" minOccurs="0"
    maxOccurs="unbounded"/>
   <xs:element ref="Virtex" minOccurs="0"
    maxOccurs="unbounded"/>
   <xs:element ref="Edge" minOccurs="0"
    maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="graphId"
   type="xs:string" use="optional"/>
  <xs:attribute name="graphType"
   type="GRAPH-TYPE" use="optional"/>
  <xs:attribute name="graphType"
   type="MODEL-TYPE" use="optional"/>
  <xs:attribute name="recordCount"
   type="NUMBER" use="optional"/>
 </xs:complexType>
</xs:element>
```

以下は上記の $graphType$ で指定する際のグラフのタイプ $GRAPH-TYPE$ を定義するスキーマである．$GRAPH-TYPE$ が文字列で示されねばならないこと，そして，元データ $original$ か，マイニング結果として導かれる誘導部分グラフ $induced$ であるか一般部分グラフ $general$ であるかのいずれかでなければならないことが指定されている．

```
<xs:simpleType name="GRAPH-TYPE">
 <xs:restriction base="string">
  <xs:enumeration value="original"/>
  <xs:enumeration value="induced"/>
  <xs:enumeration value="general"/>
 </xs:restriction>
</xs:simpleType>
```

以下は同様にグラフマイニング結果として得られる構造の種類を $MODEL-TYPE$ として定義している．ルート無しツリー，ルート付きツリー，順序木，パス，閉路を含むグラフのいずれかであること示している．

```
<xs:simpleType name="MODEL-TYPE">
  <xs:restriction base="string">
   <xs:enumeration value="unRootedTree"/>
   <xs:enumeration value="rootedTree"/>
   <xs:enumeration value="orderedTree"/>
   <xs:enumeration value="path"/>
   <xs:enumeration value="graph"/>
 </xs:restriction>
</xs:simpleType>
```

更に以下2つは，グラフの構成基本要素である$Virtex$と$Edge$を定義するスキーマである．$Virtex$も$Edge$も同様に$< xs : complexType >$タグ内の複数タグで定義される．$Virtex$は$< xs : element >$として$VirtexLabel$を参照する．また$< xs : attribute >$として$Virtex$の識別名である$virtexId$，次数$dimension$を持つ．次数によって$VirtexLabel$として3次元座標のような多次元ベクトルを持たせることが可能になる。更に下部には$VirtexLabel$の定義も与えられている．これはラベル名称$field$とその値$value$からなる．$Edge$の定義はより複雑である．同様に$EdgeLabel$定義を参照し，$edgeId$や$EdgeLabel$の$dimension$を有するが，それ以外にも有効エッジであるか無向エッジであるかを示す$graphtype$，いずれの$Virtex$間を結ぶかを示す$bgnvirtexid$，$endvirtexid$を$< xs : attribute >$で指定する．

```
<xs:element name="Virtex">
 <xs:complexType>
 <xs:sequence>
  <xs:element ref="VirtexLabel"
   minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
  <xs:attribute name="virtexId"
   type="xs:string" use="required"/>
  <xs:attribute name="dimension" type="xs:int"/>
 </xs:complexType>
</xs:element>
<xs:element name="VirtexLabel">
 <xs:attribute name="field" type="xs:string"/>
 <xs:attribute name="value" type="xs:string"/>
</xs:element>

<xs:element name="Edge">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="EdgeLabel" minOccurs="0"
     maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="edgeId"
   type="xs:string" use="required"/>
  <xs:attribute name="graphtype"
   type="EDGE-TYPE" default="undirected"/>
  <xs:attribute name="demension" type="xs:int"/>
  <xs:attribute name="bgnvirtexid"
   type="xs:string"/>
  <xs:attribute name="endvirtexid"
   type="xs:string"/>
```

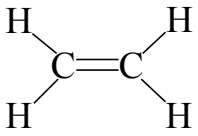

Fig. 1: エチレンの平面構造

```
 </xs:complexType>
</xs:element>

<xs:simpleType name="EDGE-TYPE">
 <xs:restriction base="string">
  <xs:enumeration value="directed"/>
  <xs:enumeration value="undirected"/>
 </xs:restriction>
</xs:simpleType>

<xs:element name="EdgeLabel">
 <xs:attribute name="field" type="xs:string"/>
 <xs:attribute name="value" type="xs:string"/>
</xs:element>
</xs:schema>
```

3.2 グラフ表現の実例

以上のスキーマに基づき，図 1 に示すエチレンをPMML 上で表現した例を以下に示す．最初の3行はXML 及び PMML のバージョン，文字コード指定，著作権明示などの情報である．次の$< DataDictionary >$タグに挟まれた部分では，属性名に$atomy$と$bondtype$があり，両者は記号属性であって，$atomy$の値はHとCのいずれかであり$bondtype$の値は単結合，芳香族結合，二重結合，三重結合のいずれかであることを記述している．その次の$< GraphMode >$タグ部分によって実際のグラフが記述されている．モデル名は仮に$sample$としてある．$< MiningSchema >$タグではこのモデルが$atomy$と$bondtype$という属性を扱うことを示しである．$< Graph >$タグ内にはエチレンが記述されている．各$< Virtex >$タグは1つの$Virtex$を表し，それぞれに識別番号が振られ，かつ$atomy$というスカラーのラベルが付与されていること ($dimension$が 1)，またそれぞれの元素名 (H または C) が記述されている．各$< Edge >$タグは1つの$Edge$を表し，それぞれに識別番号が振られ，結合には方向性がないので$graphtype$として$undirected$，$bondtype$として単結合か二重結合，始点$bgnvirtexid$及び終点$endvirtexid$の$Virtex$の識別番号が記述されている．エチレンでは二重結合は2つの炭素 C 間にしか存在しないので，対応する3番と4番の$Virtex$間の結合だけが二重結合となっている．これは1番目のグラフの記述であるが，以下続けて他の識別番号で複数のグラフを記述することが可能である．

```
<?xml version="1.0" encoding="UTF-8"?>
<PMML version="2.1" >
<Header copyright="www.dmg.org"
 description="Sample Graph Model"/>
  <DataDictionary numberOfFields="2">
   <DataField name="atomy" optype="categorical">
    <Value value="H"/>
    <Value value="C"/>
   </DataField>
   <DataField name="bondtype"
    optype="categorical">
    <Value value="singlebond"/>
    <Value value="aromaticbond"/>
    <Value value="doublebond"/>
    <Value value="triplebond"/>
   </DataField>
  </DataDictionary>
  <GraphModel modelName="sample">
   <MiningSchema>
    <MiningField name="atomy"/>
     <MiningField name="bondtype"/>
    </MiningSchema>
    <Graph graphId="1">
     <Virtex virtexId="1" dimension="1">
      <VirtexLabel field="atomy" value="H"/>
     </Virtex>
     <Virtex virtexId="2" dimension="1">
      <VirtexLabel field="atomy" value="H"/>
     </Virtex>
     <Virtex virtexId="3" dimension="1">
      <VirtexLabel field="atomy" value="C"/>
     </Virtex>
     <Virtex virtexId="4" dimension="1">
      <VirtexLabel field="atomy" value="C"/>
     </Virtex>
     <Virtex virtexId="5" dimension="1">
      <VirtexLabel field="atomy" value="H"/>
     </Virtex>
     <Virtex virtexId="6" dimension="1">
      <VirtexLabel field="atomy" value="H"/>
     </Virtex>
     <Edge edgeId="1" graphtype="undirected"
      demension="1" field="bondtype"
      value="singlebond" bgnvirtexid="1"
      endvirtexid="3"/>
     <Edge edgeId="2" graphtype="undirected"
      demension="1" field="bondtype"
      value="singlebond" bgnvirtexid="2"
      endvirtexid="3"/>
     <Edge edgeId="3" graphtype="undirected"
      demension="1" field="bondtype"
      value="doublebond" bgnvirtexid="3"
      endvirtexid="4"/>
     <Edge edgeId="4" graphtype="undirected"
      demension="1" field="bondtype"
      value="singlebond" bgnvirtexid="4"
      endvirtexid="5"/>
     <Edge edgeId="5" graphtype="undirected"
      demension="1" field="bondtype"
      value="singlebond" bgnvirtexid="4"
      endvirtexid="6"/>
    </Graph>
     . . .
  </GraphModel>
</PMML>
```

4 グラフマイニングツールへの PMML の実装

最初に述べた現状のグラフマイニングツールは，いずれも現状では上記で説明した PMML に準拠したデータの読み書きができない．例えば AGM は述語によって記述されたデータの読み込みを行うようになっている．また，掲載は省略するがマイニングよって得られた多頻度部分グラフは，隣接行列のまま出力される．更に化学分野では，化学式を SDI ファイルや Mol ファイルといった独自の形式で記述する規格が普及している．そこで現在，

1. AGM 述語形式と PMML グラフ表現間の parser
2. SDI ファイル形式と PMML グラフ表現間の parser
3. 隣接行列形式と PMML グラフ表現間の parser

のプログラム整備を進めており，(1)，(2) については既に完成している．また，AGM や GBI といったマイニングツール自身が直接 PMML ベースグラフ表現を読み書きできるように変更することを計画中である..

5 おわりに

本論では，グラフデータ構造は現状の PMML の規格が扱うデータやモデル構造よりも複雑であるにも関わらず，グラフ構造表現に必要なスキーマを現状の PMML に付加することで，十分に表現能力の高い柔軟な記述が可能であることを明らかにした．更に提案した規格が高い汎用性と柔軟性を持つため，従来各分野やグラフマイニングツール毎に個別に用いられてきた表現と提案表現間の変換プログラムである各種 parser を容易に開発できることも分かった．今後，本表現規格を一層改良，拡張していくと共に，各種 parser の準備やグラフマイニングツールへの PMML インターフェイスの実装，更にそれらの MUSASHI への実装を図る予定である．

参考文献

[1] Srikant, R., Vu, Q. and Agrawal, R.: Mining Association Rules with Item Constraints, *Proc. of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pp.67-73, (1997).

[2] Agrawal, R. and Srikant, R.: Mining Sequential Patterns, *Proc. of the 11th International Conference on Data Engineering (ICDE'95)*, pp.3-14, (1995).

[3] Mannila, H., Toivonen, H. and Verkamo, A.I.: Discovery of Frequent Episodes in Event Sequences, *Data Mining and Knowledge Discovery*, Vol. 1, No. 3, pp.259-289, (1997).

[4] Sintani, T. and Kituregawa, M.: Mining Algorithms for Sequential Patterns in Parallel: Hash Based Approach, *Proc. of the 4th International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pp.283-294, (1998).

[5] Shoudai, T. et al.: BONSAI Garden: Parallel Knowledge Discovery System for Amino Acid Sequences, *Proc. of the Third International Conference on Intelligent Systems for Molecular Biology*, AAAI Press, pp.359–366, (1995).

[6] Zaki, M.: Efficiently Mining Frequent Trees in a Forest, *Proc. of International Conference on Knowledge Discovery and Data Mining*, (2002).

[7] Asai, T. et al.: Efficient Substructure Discovery from Large Semi-Structured Data, *Proc. of SIAM International Conference on Data Mining*, pp.158—174, (2002).

[8] Asai, T. at al.: Discovering frequent substructures in large unordered trees, Tatsuya Hiroki Arimura, Takeaki Uno, and Shin-ichi Nakano, *Proc. of the 6th International Conference on Discovery Science (DS'03)*, LNAI, Springer-Verlag, (2003).

[9] Nijssen, S. and Kok, J.N.: Effcient Discovery of Frequent Unordered Trees, *Working Note of First International Workshop on Mining Graphs, Trees and Sequences (MGTS-2003)*, pp.55–64, (2003).

[10] Cook, J. and Holder, L.: Substructure discovery using minimum description length and background knowledge. *J. Artificial Intelligence Research*, Vol.1, pp.231–255, (1994).

[11] Yoshida, H., Motoda, K., and Indurkhya, N.: Graph-based induction as a unified learning framework, *J. of Applied Intelligence*, Vol.4, pp.297–328, (1994).

[12] Dehaspe, L. and Toivonen, H.: Discovery of frequent datalog patterns, *Data Mining and Knowledge Discovery*, Vol.3, No.1, pp.7–36, (1999).

[13] Nijssen, S. and Kok, J.: Faster association rules for multiple relations, *Proc. of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, Vol.2, pp.891–896, (2001).

[14] A. Inokuchi, Washio, T. and Motoda, H.: An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data, *Proc. of PKDD2000: Principles of Data Mining and Knowledge Discovery, 4th Europian Conference, Lecture notes in Artificial Intelligence 1910*, Jan Zytkow Eds., pp.13-23, (2000).

[15] A. Inokuchi, Washio, T. and Motoda, H.: Complete mining of frequent patterns from graphs: Mining graph data, *Machine Learning*, Vol.50, pp.321–354, (2003).

[16] Kuramochi, M. and Karypis, G.: Frequent subgraph discovery, *Proc. of the first IEEE Conf. Data Mining (ICDM'01)*, pp.313–320, (2001).

[17] Yan, X. and Han, J.: gspan: Graph-based substructure pattern mining, *Proc. of the second IEEE Conf. Data Mining (ICDM'02)*, pp.721–724, (2002).

[18] de Raedt, L. and Kramer, S.: The levelwise version space algorithm and its application to molecular fragment finding, *Proc. of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, Vol.2, pp.853–859, (2001).

[19] Workshop multi-relational data mining (MRDM' 01), In conjunction with PKDD'01 and ECML' 01, (2002), http://www.kiminkii.com/mrdm/.

[20] First International Workshop on Mining Graphs, Trees and Sequences (MGTS'03): In conjunction with PKDD'01 and ECML'01, (2002), http:// www.ar.sanken.osaka-u.ac.jp/MGTS-2003 CFP.html.

[21] Muggleton, S: Inverse entailment and Progol, *New Generation Computing Journal, Special issue on Inductive Logic Programming*, Vol.13, pp.245-286, (1995).

[22] Fernandez, M., Morishima, A., Suciu, D.: Efficient Evaluation of XML Middle-ware Queries, *Proc. of the ACM SIGMOD Conference on Management of Data*, (2001).

[23] 羽室行信，加藤直樹，矢田勝俊，鷲尾隆: MUSASHI でらくらくデータマイニング, Software Design, 10月号, pp.83-91, (2003).

[24] IBM Corp, KXEN, Magnify Inc., Microsoft, MINEit Software Ltd., University of Illinois, Oracle Corporation, Salford Systems, SAS Inc., SPSS Inc., StatSoft, Inc.: Data Mining Group, http://www.dmg.org/index.htm.

[25] Project: Predictive Model Markup Language (PMML), SOURCEFORGE.NET, http://sourceforge.net/projects/pmml.