例示操作に基づくXQuery問合せ学習システムのプロトタイプ開発

Development of a Prototype System for Learning XQuery Queries Based on Example Operations

森嶋 厚行♥ 中溝 昌佳♣ 北川 博之♠ 松本 明♦

Atsuyuki MORISHIMA Akiyoshi NAKAMIZO
Hiroyuki KITAGAWA Akira MATSUMOTO

既存のXML問合せ言語や操作系では，一般に，ユーザはテキストエディタや視覚的操作系を用いて問合せを記述する．XLearnerは全く異なるアプローチのXML操作系であり，ユーザの例示操作に基づきXQuery問合せを学習する．完全に規則的な構造を持つRDBとは異なり，半構造データの一種であるXMLを対象とした問合せの学習は自明ではない．本稿では特にXLearnerプロトタイプシステムの実装について説明する．

Existing XML query languages are textual or graphical languages in which we can specify queries for XML manipulation. XLearner, a different kind of manipulation framework for XML, learns XQuery queries based on operations of sample XML elements. Inferring queries for XML is a non-trivial task, because XML is a kind of semistructured data, in contrast to relational databases whose data structure is completely regular. The paper focuses on the implementation of a prototype system of XLearner.

1. はじめに

これまでにXMLを対象とした問合せ言語や操作系が数多く提案されてきた．これらでは一般に，ユーザはテキストエディタや視覚的操作系を用いて問合せを記述する．我々はこれらとは全く異なるアプローチのXML操作系であるXLearnerの研究開発を行なっている．XLearnerは，サンプルのXML 要素に対するユーザの例示操作から，XQuery問合せを学習するシステムである．本稿では，XLearnerプロトタイプシステムの実装について述べる．

XLearnerプロトタイプシステムの開発にあたっては次の2点に留意した．第1 に，ユーザとXLearnerの間で必要なインタラクション数を減らす工夫を行なうことである．第2に，XQueryの学習処理におけるコストの削減を行なうことである．本稿では，まずXLearnerの概要を説明し，次にプロトタイプシステムの実装について記述する．

♥ 正会員 筑波大学知的コミュニティ基盤研究センター
amorishima@acm.org
♣ 学生会員 芝浦工業大学大学院工学研究科
m103198@sic.shibaura-it.ac.jp
♠ 正会員 筑波大学電子・情報工学系
kitagawa@is.tsukuba.ac.jp
♦ 筑波大学大学院システム情報工学研究科

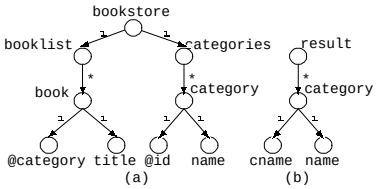

図 1 DTD

```
<result>{
for $c in /bookstore/categories/category
return <category>
       <cname>{$c/name}</cname>{
       for $b in /bookstore/booklist/book
       where $b/@category = $c/@id
       return <name>{$b/title}</name>
       }</category>
}</result>
```

図 2 XQuery 問合せ
(document 関数と text 関数は省略)

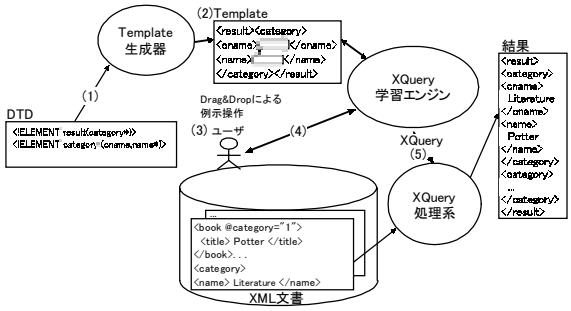

図 3 XLearner による問合せ学習の流れ

2. XLearner

まずXLearnerの利用例を説明する．図1(a) のDTD で表されるXMLデータに対する問合せを学習させたいとする．このデータは，本屋における本のリストと分野情報を表している．bookは本を表す．categoryはbookが属する分野を表す．各bookがどの分野に属するかは，bookの子要素であるcategory属性に，category要素へのIDREF参照を持つことで表す．この時，学習させたいXQuery問合せの例を図2に示す．これは，各分野毎に，分野の名前とその分野の本の名前のリストを返す問合せである．

次にXLearnerの使い方を説明する(図3)．まず，結果として欲しいXMLデータのDTD(図1(b))をテンプレート生成器に入力する．次に，ユーザは操作対象となるXML文書の中から，いくつか要素もしくは値（**XMLノード**と総称する）を選び，それらをテンプレートにドラッグアンドドロップ（以下D&D）し，欲しいXML問合せ結果の一部分を示す．テンプレートにD&DされたXMLノードを**例示ノード**と呼ぶ．

与えられたXML問合せ結果の一部分を手がかりに，学習エンジンは，ユーザが意図する問合せを学習するための質問をユーザに対して行う．このように，システムが能動的に質問を行ないながら学習を行なう枠組みを能動学習という．最後に，システムはXQuery問合せを出力し，XQuery処理系に渡す．

最初にあげた問合せ例の学習過程は次のようになる．まず，結果のDTDをテンプレートに入力すると，XLearnerは図4(b)のテンプレートを表示する．ユーザはXMLブラウザ中のXMLの要素をいくつか選び，D&Dし，図4のようにテンプレートを埋める．矢印はD&D操作を表している．

次にXLearnerは，ユーザの意図したXQuery問合せを求めるために，ユーザに対して質問を行なう．これらは，各例示ノードeの**extent**(以下EXT_e)を決定するための質問である．

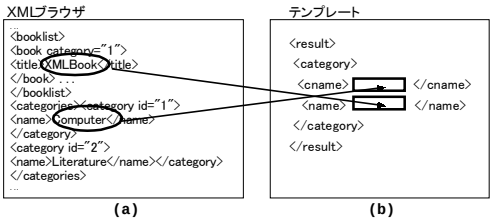

図 4 XML ブラウザとテンプレート

EXT_eは，例示要素eが与えられた**文脈**(以下$context_e$)においてeが代表しているノードの集合である．ここで$context_e$とは，EXT_eより前にextentが決定された例示ノードの集合である．例えば，“Computer”，“XMLBook”の順にextentが決まるとすると，$context_{\text{Computer}}$は空であり，$context_{\text{XMLBook}}$ = {Computer} である．その結果，$\text{EXT}_{\text{Computer}}$ は全ての分野の名前の値の集合となり，EXT_{XMLBook}は，その文脈($context_{\text{XMLBook}}$)における，bookの集合(すなわち，コンピュータ分野の本の集合)となる．

XLearnerが行なう質問には，**Membership Query(MQ)**と**Equivalence Query(EQ)**がある．MQでは，XLearnerがあるXMLノードを選び，そのノードがEXT_eに含まれるかどうかをユーザに質問する．画面上では，対象ノードの上で「Yes」「No」のポップアップメニューが表示されるので，ユーザはそのノードがEXT_eに含まれる場合は「Yes」を，含まれない場合は「No」を選ぶ．EQは，XLearnerが生成した仮説extent $E\hat{X}T_e$ に関して，$E\hat{X}T_e = EXT_e$ が成立するかを質問する．画面上で，$E\hat{X}T_e$ であるノード群がハイライト表示されるので，ユーザは，$E\hat{X}T_e = EXT_e$ ならば「OK」ボタンを押し，間違っている場合は反例を与える．反例には正の反例($\in EXT_e - E\hat{X}T_e$)と負の反例($\in E\hat{X}T_e - EXT_e$)がある．反例のノードの上で右クリックし，そのノードが正の反例ならば「Yes」を，負の反例ならば「No」を選ぶ．

3. 問合せ学習機構の概要

内部的には，XLearnerはXQuery問合せを直接学習するのではなく，XLearnerにおけるXQuery問合せの内部表現である**XQ-Tree**の学習を行なう．図5は，図2の問合せを表すXQ-Treeである．XQ-Treeの各ノードnはNi:-q(n)の形式をしている．ここで，Niはノード識別子であり，q(n)は，問合せ断片（flwr式 [for $expr_f$ [where $expr_w$]] return $expr_r$,ここで[...]は省略可)である．XQuery問合せQが与えられた時，それに対応するXQ-Tree tは次の手順で計算される：(1)let式を除去し，変数をそれに割り当てられた式で置き換える．(2)全てのXMLノードを返す式(例えば，$c/name)を，等価なflwr式（for $cn in $c/name return $cn）に書き換える．これにより入れ子のflwr式ができる．(3)以上で作成したflwr式間の入れ子関係を木構造で表す．

問合せの学習は，2つのステップから成る．

（ステップ1）XQ-Treeスケルトンの作成：与えられたDTDとD&Dで得られた例示ノードから，学習結果のXQ-Treeの基となるXQ-Treeスケルトンを生成する．図6は図1(b)のDTDと図4の操作によって生成されたXQ-Treeスケルトンである．各ノードnは，q(n)=“return <elem>$expr$</>”の形式をしている．ここで，elemはタグ名であり，$expr$は子ノードへの参照を持つ．さらに，例示ノードeがドロップされた場所がXQ-Treeノードnに対応する場所であったならば，q(n)の$expr$はeに対応する変数名v_eを持つ（この例では$cnと$n）．

```
N1:-     return <result>N1.1</result>
N1.1:-   for $c in /bookstore/categories/category
         return <category>N1.1.1 N1.1.2</category>
N1.1.1:- for $cn in $c/name return <cname>$cn</cname>
N1.1.2:- for $b in /bookstore/booklist/book, $n in $b/name
         where some $bc in $b/@category satisfies
               (some $ci in $c/@id satisfies $bc=$ci)
         return <name>$n</name>
```

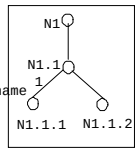

図 5 XQ-Tree 表現

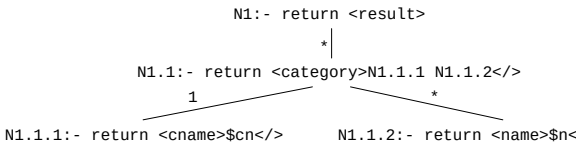

図 6 XQ-Tree スケルトン

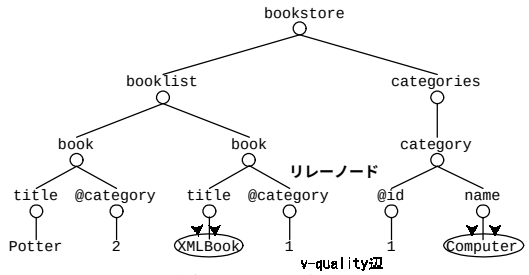

図 7　XML グラフ

（ステップ2）各問合せ断片の学習：XQ-Treeスケルトンを走査しながら，例示要素eをドロップされたXQ-Treeノードn_eのそれぞれに対して，各問合せ断片q(n_e)=for $expr_f$ where $expr_w$ return $expr_r$の学習をしていく．実際にXLearnerが学習可能なクラスでは，これらの形式は限定されている[4].特に，$expr_r$はXQ-Treeスケルトンで与えられた式そのものに限定されているので，実際に学習するのは，$expr_f$と$expr_w$である．これらはEXT_eに関するMQとEQの結果から計算される[4]．それぞれについての概要を次に説明する．

(1) $expr_f$の学習：例えば，q(N1.1)では，$expr_f$= $c in /bookstore/categories/categoryを学習する必要がある．これは，変数$cにXPath式で表されるXMLノードを束縛させるための式である．XML要素のパスを文字列と見なすと，XPath式を学習することは，本質的にはパスの集合が表す言語を受理する有限オートマトンを求めることに対応する．ある言語が与えられた時，それを受理する最小の有限オートマトンを見つけることはNP完全[3]である．一方，能動学習を利用したAnguinのオートマトン導出アルゴリズム[1]を利用すれば，多項式時間で発見することが可能である．XLearnerでは，Anguinのオートマトン導出アルゴリズムを利用してEXT_eに関するMQとEQからパス式の学習を行う[4]．

(2)$expr_w$の学習：例えば，q(N1.1.2)では，$expr_w$=some $bc in $b/@category satisfies (some $ci in$c/@id satisfies $bc=$ci)を学習する必要がある．XLearnerは$context_e$中のXMLノード(集合)と例示ノードeに関して成立する述語を全て数え上げ，その積を$expr_w$の候補として計算する．実際には，いくつかの述語が除去されて最終的な$expr_w$となる[4].述語を全て数え上げるのに利用するのが，XMLグラフである(図7)．これは，XQuery and XPath data model[8]等と同様に，XMLインスタンスの構造を木構造で表したものである．さらに，我々のXMLグラフでは，**v-equality辺**を持つ．これは，同じ値を持つXMLノード間に存在する辺である．$expr_w$は次のように求められる．（1）XMLグラフにおいて，XMLノード集合{e}∪$context_e$に含まれる任意の2ノード間のパスを全て求める．(図7は，e=XMLBook，$context_{\text{XMLBook}}$={Computer}の場合を示す)

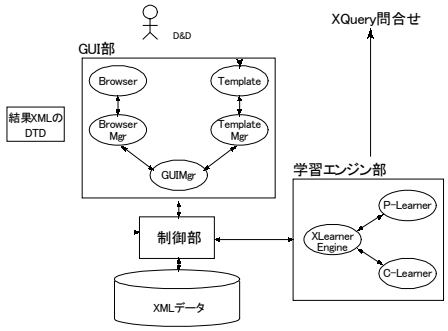

図 8 プロトタイプシステムのアーキテクチャ

この処理は，一般には指数関数オーダの処理が必要である．

（2）各パスに関して成立する制約（述語）を数え上げる．

この述語の種類はXLearnerの仕様により限定されており[4]，数え上げが可能である．その結果，q(N1.1.2)の$expr_w$が得られる．

4. プロトタイプシステムの実装

本プロトタイプシステムの実装にはJava(Java2 SDK 1.4)を用いた．GUI部の実装にはSwingおよびDrag and Drop APIを利用した．システム構成図を図8に示す．プロトタイプシステムは制御部を中心にGUI部と学習エンジン部によって構成される．XLearnerは，能動学習を行なうため，学習エンジンが能動的に動く必要がある．したがって，GUI部と学習エンジン部はそれぞれ別スレッドで動作する．

現段階のシステムには次の制約がある．まず，XMLブラウザからテンプレートにD&D可能なものは，末端のXMLノードに限定される．また，再帰的定義を持つDTDは，結果のXMLが従うDTDとして受け付けることができない．

GUI部: Browserは，XMLブラウザを実装したモジュールである．操作対象のXMLを表示する役割や，MQ，EQの質問をユーザに示し，質問に対するユーザの答えを受け取る役割を果たす．Templateは，テンプレートを実装したモジュールである．テンプレート表示を行い，ユーザのD&D操作によって例示ノードを受け取る．BrowserMgr， TemplateMgrはそれぞれXMLブラウザおよびテンプレートを管理する．GUIMgrはGUI全体を管理するほか，D&Dされた例示要素とテンプレートに基づき，XQ-Treeスケルトンを作成する役割を果たす．

学習エンジン部: 学習エンジン部の役割は，GUIMgrより作成されたXQ-Treeスケルトンを走査しながら，各ノードの問合せ断片を学習することである．学習エンジン部は大きくわけて，XLearnerEngine，P-Learner，C-Learnerの3つで構成される．XLearnerEngineは，XQ-Treeスケルトンを走査するアルゴリズムを実装している．各ノードの問合せ断片の学習は，P-LearnerとC-Learnerが行う．

P-Learnerは，Angluinのオートマトン導出アルゴリズムを用いて各問合せ断片の$expr_f$を学習する．図9にP-Learnerの内部構造を示す．P-Learnerは有限オートマトン学習部，インタラクション削減部，MQ対象ノード選択部，オートマトン-パス正規表現変換部から構成されている．有限オートマトン学習部は，XLearnerEngineから例示ノードeを受け取り，これを手がかりにオートマトン学習を行なう．有限オートマトン学習部は，学習の過程でユーザに対してMQを発行する．詳細は4.2節で説明するが，実はユーザに質問せずに処理できるMQが存在する．この判断を行なうのがインタラクション削減部である．MQ対象ノード選択部は，実際にMQを行なう要素を選択するモジュールである．この選択処理が必要な理由は

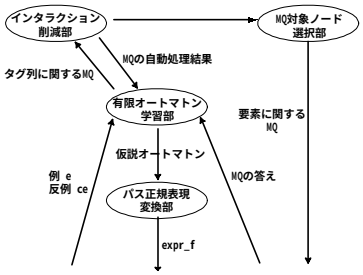

図 9 P-Learner の内部構造

次の通りである．XLearnerの文脈では，Angluinのアルゴリズムにおける(オートマトンが受理する)文字列に対応するものは， XML文書のルート要素から特定の要素にいたるタグ列になる．したがって，Angluinのアルゴリズムをそのまま用いると，MQはタグ列に関して発行されてしまい，特定の要素に対するMQとはならない．言い換えると次のようになる．まず，path(e)をXMLのルート要素から要素eへの要素の列とする．また，tags(e_p)をe_pにマッチするタグ列とする．このとき，一般に，あるタグ列 ts が与えられた時， ts=tags(path(e)) を満たすeは複数存在する．このように，タグ列と特定の要素は一対一対応しないため，XLearnerの文脈でMQを行なう際にはタグ列にマッチする要素の中から一つの要素を選択する必要がある．本プロトタイプシステムでどのような基準で要素を選択するかは，4.2節で説明する．最後に，パス正規表現変換部は，有限オートマトン学習部が生成したオートマトンをパス正規表現に変換する．

C-Learnerは $expr_w$ の学習を担当する．これは3章で述べたように，XMLグラフにおける例示ノード間のパスで成立する述語を数え上げることにより行われる．C-Learnerの主要なモジュールはcond関数とrelax関数である．

cond関数: この関数は引数としてXMLグラフにおけるノードの集合を取る．返り値は，各ノード間のパスに関して成立する全ての述語の集合である．

relax関数: この関数は引数として述語集合の組(u,v)を取る．戻り値は，直感的にはuとvの積集合である(名前の不一致を解決してから行なうので，厳密には異なる．)

4.1 処理の流れ

処理の流れは次のようになる．システムを立ち上げると，Browserが操作対象となるXMLデータを表示する．ユーザが，問合せ結果が従うべきDTDをシステムに与えると，TemplateMgrがテンプレートを作成し，Templateに表示させる．この状態で，システムはユーザのD&D操作を待つ．ユーザはBrowserに表示されているXMLノードをTemplateにD&Dし，Templateを埋める．D&D操作が完了したら．GUIMgrがXQ-Treeスケルトンを作成する．XLearnerEngineはXQ-Treeスケルトンを受け取り，学習を始める．各問合せ断片の学習で行なわれるユーザとのインタラクション(MQやEQ)の結果は，P-LearnerおよびC-Learnerに渡される．P-LearnerおよびC-Learnerによって出力された$expr_f$および$expr_w$を元に仮説flwr式が作成され，EQが行なわれる．反例ceが与えられた際には，有限オートマトン学習部はそのceを基にオートマトンの学習を続ける．システムがMQやEQを行なう際には，学習エンジンがBrowserにその質問を表示させるよう指示する．

図10はプロトタイプシステムの実行画面である．左側がXMLブラウザ，右側がテンプレートである．

4.2 実装における問題と解決のためのアプローチ

XLearnerのナイーブな実装には，次のような問題点がある．

(1) 学習に必要なインタラクション数の問題: n, k, m をそ

図 10 プロトタイプシステムの実行画面

れぞれ，学習する有限オートマトンの状態数，アルファベットの数(すなわちタグの種類の数)，反例の最大のサイズ(XMLインスタンスにおけるルートノードからXMLノードへの長さ)とする．このとき，インタラクションの数は$O(kmn^2)$である．すなわち，簡単な有限オートマトンの学習でさえ，数百回のインタラクションが必要な可能性がある．

XLearnerは，次に示す2つのルールでMQを削減する[4]．

R1. DTDに矛盾するタグの並びのMQに対し自動的に「No」を与える．

R2. 例示ノードeが与えられたときにeのパスpath(e)の末端のタグが$t1$だとする．この時，タグの並びの末端が$t1$でないMQに対し自動的に「No」を与える．これは，パターンマッチングにおいて，末端のタグが重要であるというヒューリスティクスに基づいている．

さらに本プロトタイプシステムでは，次のルールを組み込む．

R3. MQ対象となるXMLノードの候補が複数ある場合は，それらのうち，XMLグラフ上で例示ノードeに最も近いノードを優先してMQを行なう．これは，eに近いノードはEXT_eに含まれる(すなわち「Yes」が返される)可能性が高いというヒューリスティクスに基づく．XLearnerの学習アルゴリズムは，MQに対して「Yes」が返された時はバックトラックを起こさないという性質がある[4]ため，インタラクション数を減らす効果がある．

(2) $expr_w$の学習コストの問題:2つの問題がある．第1に，全てのv-equality辺を保持しておくには多くの余分な記憶容量が必要となる．第2に，例示ノード間の全てのパスを数え上げることは探索コストが高い．本プロトタイプシステムはヒューリスティックスと既存技術を応用し，これらに対処する．

v-equality辺の問題.v-equality辺は問合せに含まれる結合条件を導出するために利用されるが，実際に結合条件に用いられる値の組合せは限られている．例えば，XMark[6]とXML Use Case[7]では，27の等結合のうち3つは明示的なidref-id参照である．残りは要素の値を用いているが，これらはすべて外部キーの役割を果たす値である．このような値に対してはインデックスが存在することが一般的であると考えられる．本プロトタイプシステムではこのようなインデックスの存在を仮定し，これを利用して(論理的な)XMLグラフの探索を行なう．特別にv-equality辺を表すためのデータを持つことはしない．

パスの計算コスト.一般的なXQuery問合せにおいては，結合に利用するXMLノード間の(XMLグラフにおける)距離はそれほど長くない．例えば，XMarkとXML Use Casesにおける最大の長さは12であるため，XMLノード間のパスを計算するためには，それぞれのXMLノードから6ずつ探索すればよい．それに加え，Graph Schemas[2]のようなグラフ構造のメタデータを利用し，探索空間をあらかじめ制限する．

(3) EQ処理を行なう際のパフォーマンスの問題： 2章で説明したとおり，XLearnerがEQを質問するためには仮説extent $\hat{EXT}$ を計算する必要がある．この計算は一般に処理時間がかかる．本システムでは，Browser中に一度に表示できるXMLデータは全体のごく一部である点に着目し，$\hat{EXT}$ の計算をXMLデータ全体に対して行うのではなく，Browserに表示されている範囲のデータに対してのみ行なう．これは[7]と同様の手法である．ただし現時点ではこの処理は未実装である．

5. まとめ

本稿では，XQuery問合せの学習システムXLearnerの概要および，プロトタイプシステムの実装について述べた．今後の課題としては，ユーザビリティーの検証が挙げられる．

[謝辞]

本研究の一部は，日本学術振興会科学研究費基盤研究(B)(12480067) による．

[文献]

[1] D.Angluin. Learning regular sets from queries and counterexamples. Information and Conputation, 75(2): 87-106, 1987

[2] P.Buneman, S.Davidson, M.Fernandez, and D.Suciu. Adding structure to unstructured data. Proc. ICDT, pp.336-350, 1997.

[3] E.M.Gold, Complexity of automaton identification from given data. Information and Control 37:302-320, 1978

[4] A.Morishima, A.Matsumoto, H.Kitagawa. XLearner: A System to Learn XML Queries through Examples (投稿中).

[5] A.Morishima, T.Mouri, H.Kitagawa. An Exampled-Based Web-Site Cnstruction Tool and Its Implementation. Information Systems and Databases, pp.136-141, 2002

[6] A.Schmidt, F.Waas, M.Kersten, D.Florescu, M.Carey, I.Manolescu, and R.Busse. Why And How To Benchmark XML Databases. SIGMOD Record, 30(3):27–32, 2001.

[7] XML Query Use Cases. http://www.w3.org/TR/xmlqueryuse-cases.

[8] W3C. XQuery 1.0 and XPath 2.0 Data Model. http://www.w3.org/TR/query-datamodel/.

森嶋 厚行 Atsuyuki MORISHIMA

筑波大学知的コミュニティ基盤研究センター・図書館情報学系助教授．1998 年筑波大学大学院工学研究科修了．博士(工学)． XML 処理・管理，異種情報源統合，情報システムのためのユーザインターフェースなどに興味を持つ．ACM，IEEE-CS，情報処理学会，電子情報通信学会各会員．

中溝 昌佳 Akiyoshi NAKAMIZO

芝浦工業大学大学院工学研究科在学中. 2003 年芝浦工業大学工業経営学科卒業．

北川 博之 Hiroyuki KITAGAWA

筑波大学電子・情報工学系教授．1980 年東京大学大学院理学系研究科修了．理学博士．異種情報源統合，文書データベース，WWW の高度利用等の研究に従事．著書「データベースシステム」（昭晃堂）等．ACM，IEEE-CS，情報処理学会，電子情報通信学会，日本ソフトウェア科学会，各会員．

松本 明 Akira MATSUMOTO

2003 年筑波大学大学院システム情報工学研究科修士修了．