表形式データからのオントロジーの構築
Ontology Generation from Table Form Data

田仲正弘[*1]
Masahiro Tanaka

石田亨[*1]
Toru Ishida

[*1] 京都大学情報学研究科社会情報学専攻
Department of Social Informatics, Kyoto University

あらまし：現在の Web には利用可能なメタデータが少ないことが Semantic Web 実現にあたっての大きな問題点となっている．そのため大量の既存のデータから自動的にメタデータを生成することが重要になる．本研究では表形式データから表構造のセマンティクスを学習することで，表中のデータに関するオントロジーを構築する手法を提案する．Web から得られた表形式データに対し提案手法を適用した結果，高い精度で表に記載されているデータに関するオントロジーが得られた．

1. はじめに

Semantic Web の実現のためには大量のメタデータが必要になる．しかし人手によるアノテーションには多大なコストがかかるため，大量のデータを扱うには適していない．そこで既存のデータからメタデータを自動的に生成する必要がある．

自動的なアノテーションの研究には，自然言語で自由に書かれたテキストを対象とするもの[1]や，Web ページのレイアウト情報を利用するもの[2]などがある．本研究では，ある程度決まった構造を持っている表形式データから，オントロジーを自動的に構築する手法を提案する．Web では表形式データはオンラインショップのカタログ等に広く用いられており，自動的なアノテーションにより表中のデータに関するオントロジーが構築できれば，希望の商品を検索するなどの用途に利用できる．

表形式データからのオントロジーの構築を行う既存の研究には，表中のデータの類似度から表の構造を判断するものが多い．[3]では，大量の HTML データから表形式データを取り出し，セル中のデータが数値か文字列かなどの形式的な特徴の類似に注目して行または列から属性と属性値のペアを得る．[4]では類似するデータの配置によって表構造を解釈し，さらにデータの内容に基づいて似た情報が記述されている表を統合する．表中で類似するデータのまとまりを判断するのに知識ベースを用いる研究もあり，[5]では地名などの情報を持つ知識ベースを利用して表形式データを解析し，人手で作成した地理情報の小さなオントロジーを拡張する例を示している．また[6]では，Hurst による表の認知モデル[7]に沿ったアルゴリズムにより，表形式データから F-Logic フレームを獲得し，人手で作成されたものと比較して評価している．

いっぽう Web に存在するさまざまな表形式データからオントロジーを構築する際には，以下のような点が重要になると考えられる．

- 表の内容に依存しない
  表構造の解釈に表中のデータの特徴を利用する方法は，データの内容によっては利用できないことがある．また大きな知識ベースの構築にはコストがかかり，異なるドメインへの適用が難しくなるため，表中のデータの内容によらず表構造を解釈できる必要がある．
- 多様な関係が獲得できる
  表形式データでは，クラス-インスタンスの関係やクラスの階層関係・プロパティの階層関係などが表現されており，属性と属性値のペアなどに限定せず，多様な関係を獲得できる必要がある．

以上の点から，本研究では，表中の少数のセルについて，それらのデータの関係の記述を前提知識として与えることで，セルの形状に基づく表構造とそのセマンティクス（表構造の表現するデータ間の関係）の対応を獲得する手法を提案する．セマンティクスがわかっている表構造が表中で現れる部分からは，そこに記述されたデータに関するオントロジーが得られる．表の構造に基づいてデータの関係を得るため，データ値の形式に関するヒューリスティクスを与えておく必要がなく，また含まれているデータ値の違いによる影響を受けない．対象ドメインに関する大きな知識ベースも必要としないため，容易にさまざまなドメインの表形式データに適用できる．表構造にどのようなセマンティクスが対応づけられるかは，最初に与える前提知識に従って決定されることになる．表中に記述されている全てのデータの関係が得られない場合には，与える前提知識を増やすことで，より多くのデータの関係が得られるようになる．前提知識として表中のデータに関する記述を与えることは困難ではない．

以降では，第 2 章で表形式データからのオントロジー獲得の基本的なアイデアについて述べる．次に第 3 章で，表構造の形式化について述べる．第 4 章では，表形式データからのオントロジー獲得のアルゴリズムについて説明する．第 5 章で，提案手法の適用の結果得られたオントロジーについて評価・考察し，第 6 章で結論を述べる．

連絡先：田仲正弘，京都大学情報学研究科社会情報学専攻，
〒606-8501 京都市左京区吉田本町，TEL 075-753-5396，
FAX 075-753-4820，mtanaka@kuis.kyoto-u.ac.jp

2. 表構造の観察

表形式データでは，表の構造によってセル中のデータの関係が表されることが多い．例えば表 1 は，「一つの列はインスタンスの一つのプロパティに対応し，二行目にプロパティ名が記述され，その下にプロパティ値を記述される」という形式をとっている．このような特定の表構造とそのセマンティクスの対応が得られれば，表中で同じ構造を持つ部分に含まれる多数のデータについても，それらの関係が得られる．

表構造とセマンティクスの対応を得るには，関係が既知である表中の複数のセルのデータについて，それらのセルを含む部分の表構造の特徴を調べる．

表 1 インスタンスのプロパティを記述した表

Processor		
Product ID	Product Name	Price
P4_340	Pentium 4 3.40E GHz	$260
P4_280	Pentium 4 2.80A GHz	$140
A64_320	Athlon 64 3200+	$160

例えば表 1 において，“P4_280”が“Product ID”というプロパティの値であることが既知であるとする．このとき“P4_280”と“Product ID”のセルに関する表構造の特徴として，“Product ID”が表の二行目におかれて上の辺で幅の広いセルに隣接しており，“P4_280”は同じ列の下の部分におかれていることが挙げられる．ここで，表形式データでは多くの場合同じ特徴を持ったセルの並びに同じ種類のデータが記述されることを考慮すると，「上に幅の広いセルが隣接しているセルにプロパティ名を記述し，同じ列の下にあるセルにプロパティ値を記述する」という，表 1 における表構造とそのセマンティクスの対応が得られる．これにより，“P4_340”や“A64_320”もプロパティ“Product ID”の値であること，“Pentium 4 3.40E GHz”，“Pentium 4 2.80A GHz”などがプロパティ“Product Name”の値であることなどがわかる．

本研究では表中のデータの関係（クラス-インスタンス関係やクラスの階層関係・プロパティ関係など）を記述する RDF ステートメントの集合を与えることで，表構造とそのセマンティクスとの対応を得る．表構造とそのセマンティクスとの対応が得られると，表中で同じ構造が出てくる箇所では，その構造に含まれるセルのデータについて関係の記述が得られる．ただし表中のより多くの箇所からセルのデータ間の関係の記述を得るには，得られた表構造をその特徴に基づいて一般化する必要がある．以下に表構造を一般化するために用いる，表形式の特徴について述べる．

1. 2 つのセルの関係は，それらのセルを含む行や列の構造で決まる

プロパティ名とそのプロパティの値など，関係する 2 つのセルは，ふつう同じ行や列にある．そのため，2 つのセルの関係は，それらを含む行や列の構造とその中でのセルの位置によって表されると考えられる．

2. 行や列の構造は，隣接するセルの大小関係や，内部の繰り返し構造で特徴づけられる

表には複数の行や列にまたがる幅の広いセルが含まれることがある．隣接する 2 つのセルで幅が異なる場合，それらのセルにはふつう異なる種類のデータが含まれており，同じ幅のセルが隣接する構造と区別する必要がある．そのため，行や列の構造は，行や列に含まれるセルとその周囲のセルとの幅の大小関係によって特徴付けられる．また，同じ行や列内で周囲のセルとの隣接関係が同じセルが連続して出現する場合には，それらのセルは出現回数によらず似た種類のデータを表していると考えられる．これは単一のセルではなく，複数のセルのまとまりに注目した場合も同様である．同じ行や列で，周囲のセルとの隣接関係が同じセルが一定のパターンで現れて特徴的なセルのまとまりを構成している場合には，それらのまとまりも出現回数によらず同じ種類の内容を表していると考えられる．

3. 3 つ以上のセルの関係は，2 つのセルの関係の組み合わせで表現できる

表中で，3 つ以上のセルが互いに関連を持つ場合がある．それらのセルのうち，2 つのセルの関係がそれらを含む行や列の構造で表現されるとき，3 つ以上のセルの関係に対応する構造は，同じ行や列にある 2 つのセルの関係を表す構造を複数組み合わせることで表現できる．関連する全てのセルが同じ行や列にある必要はなく，互いに関連を持つ 3 つ以上のセルが異なる行や列にある場合には，同じ行や列に含まれるセルを 2 つずつ選び，それらを含む行や列の構造を組み合わせて表現する．

以上の仮定に基づいて，与えた RDF ステートメントに対応する表中の複数のセルの関連を表現できるような表構造の表現を定義する．

3. 表構造の形式化

表中のデータに関する関係の記述を与えることで表構造に対してセマンティクスを対応付け，同じ表構造があらわれる部分からそこに含まれるデータに関するオントロジーを得るには，表構造が一致しているかどうかを判断できるような表構造の形式的な表現が必要になる．

表形式データにおいては全く同じ表構造ではなくても同じ意味を表す場合があり，表中のより多くの箇所からセルのデータ間の関係の記述を得るには，一般化された表現を用いる必要がある．そのため，第 2 章で述べた表形式の記述に関する仮定に基づき，以下に述べるように表構造のパターンの表現を定義する．

まず元の表の一つのセルに相当するものとして，ボックスと呼ぶ要素を定義する．さらにボックスに対応するセルの周辺のセルとの辺の共有の仕方によって，他のボックスとの関係を，図 1 のように一方向または双方向のボックス間の接続としてあらわす．

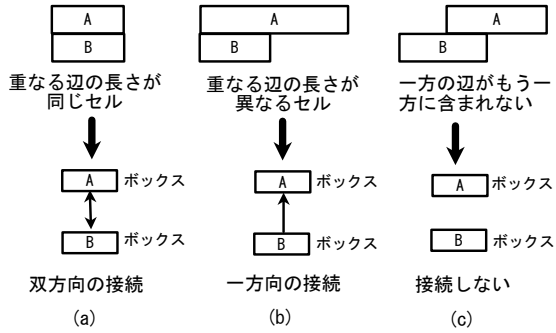


図 1 セルの辺の重なり方と対応するボックスの接続

図 1 の上部は元の表の隣接する 2 つのセルを示している. 図 1 の下部はそれらのセルに対応するボックスとそれらの関係を表している. 四角はボックスを表し, 上下に並んだ二つのボックス間の接続は, 対応するセルの隣接のしかたを表す. このセルの隣接の表現は [8]をもとにしており, このボックス間の接続を, ボックスの隣接関係と呼ぶものとする.

図 1 (a)のように, 隣接する二つのセルで重なっている辺の長さが同じ場合には, 二つのボックスの隣接関係を, 双方向の矢印で接続して表す. 図 1(b)のように, 一方の辺がもう一方の辺に含まれる場合には, 短い辺を持つセルに相当するボックスから長い辺を持つセルに, 一方向の矢印で接続して表す. 図 1 (c)のように, 隣接するセルの重なっている辺の一方が, もう一方に含まれない場合には, セルに相当するボックスは接続しない. また, 表形式データにおいて隣接する 2 つのセルのどちらが上(左)でどちらが下(右)かということは重要であるため, 矢印の上下左右の向きも区別する必要がある. これより, ボックスの隣接関係は水平であるか垂直であるかという区別と, 一方向か双方向かという区別により定義される.

この表現を用いて, 2 つのセルの関係に対応する表構造を表す. 第 2 章で述べた表構造による表現の仮定より, 行や列から 2 つのセルを両端とするセルの配列を取り出し, それをボックスとその隣接関係で表現する.

表 2 において, A, B の二つのセルが何らかの関係を持つことが既知であるとする. このとき, A, B の関係に対応する表構造を得るために, 表 2 の網掛けして示した列の構造を調べる. この列の構造をボックスとその隣接関係で表現すると, 図 2 のようになる. 各ボックスでは, 取り出した列に含まれないボックスとの隣接関係も考慮し, ボックスに接続された矢印で表す.

ここで, セル A,B に対応するボックスを, 表構造の中で何らかの役割を割り当てられているという意味で, F-ボックス (functional-box)と呼ぶ.

表 2 幅の違うセルを含む列を持つ表

	A			
		B		

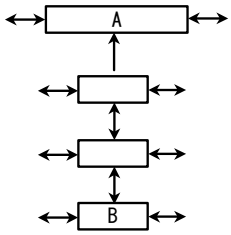


図 2 表 2 の網掛けした部分の構造

さらに, 行(列)で連続して出現するボックスが同じ隣接関係で接続され, かつそれぞれのボックスで隣接する行(列)のボックスとの隣接関係が同じ場合には, その出現回数によらずそれらの役割は同じものとみなす. ここでは, 同じ構造の繰り返しを表す＋記号(およびボックス間の接続を表す矢印・ボックスを囲む括弧)を導入することで, 連続して同じ隣接関係をもつボックスが出現する構造を表現する. ただし, A,B のセルに対応する F-ボックスについては, 同じ隣接関係を持つ場合でも同じ繰り返しの中には含めないものとする. これは, A, B の関係についての記述が与えられていることで, A,B のセルに記述されるデータはその種類を区別する必要があると考えられるためである.

図 2 で連続して現れる, 同じ隣接関係を持つボックスを＋記号を用いて表すと, 図 3 のようになる.

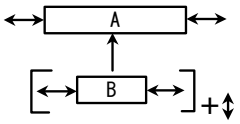


図 3 表 2 の網掛けした部分の表構造パターン

このようなボックスとその隣接関係・＋記号による表構造の表現を, 表構造パターンと呼ぶものとする.

このような表構造パターンは, 3 つ以上のセルの関係についても考えることができる. まず表中の二つのセル A, B について, それらを含む行や列の構造を得る. 次に, B と C について, それらを含む行や列の構造を得る. A,B,C についての構造を得るには, A,B と B,C それぞれについて得られた構造を組み合わせればよい.

ただし, 列の構造と行の構造を組み合わせる場合には, ＋記号による繰り返し構造の表現をどちらかに限る. これは複数の行や列での繰り返しを考えると, 行と列の構造が接続されている部分での対応関係が失われるためである.

表 3 は, 表中に A,B,C という互いに関係するセルを含む表であるとする. A,B,C の関係を表す構造は表 3 の網掛けした部分であり, ボックスとその隣接関係で表すと図 4 が得られる.

表 3 3 つのセルが互いに関係を持つ表

			B	
A			C	

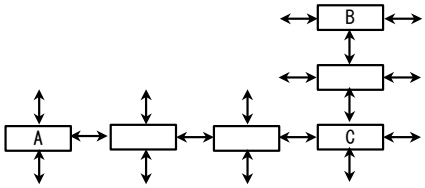
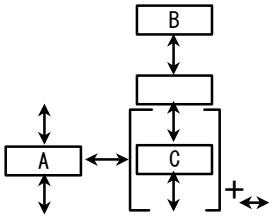
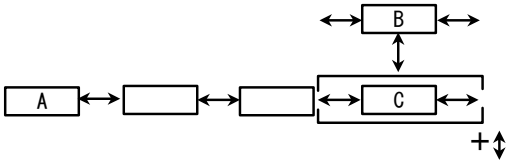


図 4 表 3 の網掛けした部分の構造

図 4 は，繰り返しとして表現される箇所をどこにとるかによって，図 5(a)(b)の 2 通りの表構造パターンが得られる．



(a)



(b)

図 5 表 3 の網掛けした部分の表構造パターン

表によっては＋記号による繰り返しのネストが現れることがある．表 4 において，A, B, C, D, E の関係に対応する表構造パターンは図 6 のようになる．

表 4 階層的な構造を持つ表

A		
B		
C	D	E

しかし図 6 の表構造パターンは，表 4 に含まれるデータの関係を全て獲得するには不十分である．なぜなら図 6 の表構造パターンには A と B の間に，B, C, D, E のボックスによって構成される部分とは異なる構造の繰り返しを挟んでいる．＋記号は一度以上の繰り返しを表すため，A のすぐ下に位置する分類とそこに含まれるインスタンスは図 6 の表構造パターンでは得られない．これは表構造パターンを A～E を含む行や列に注目して表現しており，C, D, E を含む行の構造と，A, B の間に含まれる 3 つのセルからなる行の構造を同じとみなせないためである．A のすぐ下の分類とそこに含まれるインスタンスを獲得するためには，別の表構造パターンが必要となる．

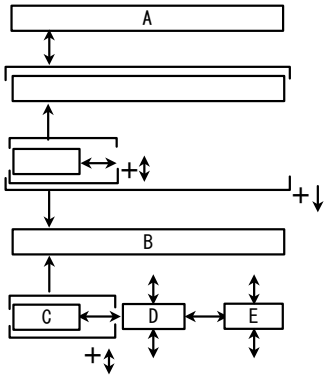


図 6 表 4 から得られる表構造パターン

4. 表構造の抽出アルゴリズム

表構造パターンとそのセマンティクスの対応を得るには，初めに表中のデータについてそれらの関係の記述を与える必要がある．本研究ではデータ間の関係を RDF ステートメントによって記述するものとする．しかしさまざまな表構造に対応する関係は，単一の RDF ステートメントでは表現できないことがある．そこで，一つ以上の RDF ステートメントの組による関係の記述をエピソードと呼び，構造パターンの獲得やあらたな関係の記述の獲得は，このエピソードの単位で行う．

表からのオントロジーの構築は，表形式データに対して以下の 1.-4.の手順で処理を適用することで行う．

Step 1. 表構造パターンの獲得

Step 2. 利用できない表構造パターンの除去

Step 3. パターンマッチによるエピソードの獲得

Step 4. 矛盾するエピソードの除去

Step 1 では，表中のデータの関係を記述したエピソードを与えることで，エピソードに記述された関係に対応する表構造パターンを得る．エピソードから表構造パターンを得る手順は以下のようになる．まず与えたエピソードに含まれる RDF ステートメントのリソースやプロパティが表中で出現する箇所を探す．次に見つかったリソースやプロパティを含む行や列の構造から，それらのリソースやプロパティが現れるセルを F-ボックスとするパターンを得る．ただし subClassOf や instanceOf のような基本的なプロパティは，プロパティ名が表の中に現れるわけではなく，表の構造それ自体によって表現される．そのため，subClassOf や instanceOf といったプロパティは，表中で現れるセルを探さず，表構造パターンに F-ボックスとして含めないものとする．

Step 2 では，Step 1 で得られたパターンのうち新たなエピソードの獲得に利用できないものを除く．与えられたエピソードを元に表構造パターンの獲得を行った結果，一般的過ぎてどのような箇所にもマッチする表構造パターンが得られることがある．例えば，表構造パターン中の全てのボックスでその隣接関係が同じ場合には，そのパターンが特定の関係に対応する表構造を表現しているとは言えない．このような表構造パターンは，与えたエピソードがあらわす関係が，表中で特徴的な表構造と対応していない場合に得られる．また，与えたエピソードに対して表構造パターンの繰り返し構造があらわす 1 対多の関係が誤っている表構造パターンが得られることがある．与えたエピソードのプロパティ値に対応する F-ボックスを一つもち，繰り返しの部分にそのプロパティ値のプロパティに対応する F-ボックスを含む表構造パターンなどがそれである．これは，繰り返し構造をどこにとるかによって一つのエピソードから複数の表構造パターンが得られることが原因である．

Step 3 では得られた表構造パターンが表中でマッチする箇所を探す．表構造パターンの各 F-ボックスにマッチするセルのデータは，表構造パターン獲得に用いた元のエピソードの対応するリソースやプロパティと同じ関係にあると考えられる．

例えばエピソードとして，リソース r_1, r_2 とプロパティ instanceOf からなる RDF ステートメントが与えられたとする．このとき表が一つの行や列の中に r_1, r_2 を値として持つセル A, B を含んでいれば，A,B を F-ボックスとして持つ表構造パターンが得られる．得られた表構造パターンが表中の別の箇所にマッチしたとすると，表構造パターンの F-ボックスの部分にマッチするセルのペア A',B' 中のデータ r_1', r_2'の関係は，r_1, r_2 の関係と同様 instanceOf であると考えられる．得られた表構造パターンが繰り返しを表す＋記号を含んでいる場合には，繰り返しの回数を変えてマッチするセルのペアを探すことにより，互いに instanceOf の関係にあるリソースのペアが得られる．

同様の考え方で，リソースだけでなく新たなプロパティも獲得することができる．エピソードとして，リソース r_1, r_2 とそれらの関係を表すプロパティ P からなる RDF ステートメントが与えられたとする．このときそれぞれ r_1, r_2, P を値として持つ 3 つのセルに関する表構造パターンが得られたならば，表構造パターン中の繰り返し回数を変えたり，表中でパターンを適用する箇所を変えてマッチするセルの組を探すことで，新たにリソースとプロパティの 3 つ組が得られる．

Step 4 では，既知のエピソードや新たに得られたエピソードの間に互いに矛盾する記述がある場合にそれらを除く．例えばパターンマッチの結果，同じ語に対してそれがプロパティ名であるという記述とクラス名であるというエピソードが同時に得られることがある．矛盾するエピソードが得られた原因は，パターン獲得に用いたエピソードに問題があった可能性があるので，パターン獲得に用いたエピソードも除く．

以上の処理の結果得られたエピソードを用いると，表中で同じリソースやプロパティが記載されている部分から新たな表構造パターンが得られることがある．そのため，Step 1～ Step 4 の処理のあと，Step 1 の処理に戻り，エピソードの集合が変化しなくなるまで処理を繰り返す．

以上の処理を行うアルゴリズム ExtractEpisodes を図 7 に示す．

```
ExtractEpisodes (Table, E)
  E' := {}
  while E <> E'
    E := E'
    for (E のそれぞれの要素 e について)
     P := make-pattern (Table, e)
    P := P / useless-pattern(P)
    for (P のそれぞれの要素 p について)
     E' := E ∪ pattern-match(Table , p)
    E' := E' / inconsistent-episodes (E')
  return E'
```

図 7 エピソード獲得のアルゴリズム

ExtractEpisodes は表形式データ *Table* とエピソードの初期集合 E を受け取る．初期状態の E は人手によって作成する必要がある．まず初めに，与えられた E の全ての要素に関して，パターン獲得を行う．make-pattern は表形式データと一つのエピソードを受け取り，得られた表構造パターンの集合を返す．得られた表構造パターンの集合を P とする．さらに，P から新たなエピソードの獲得に利用できない表構造パターンを除く．useless-pattern は表構造パターンの集合を受け取り，受け取った表構造パターンの集合のうち，一般的過ぎる表構造パターンや，誤った 1 対多の繰り返し構造を含む表構造パターンの集合を返す．次に，P の全ての要素についてパターンマッチを行って *Table* から新たなエピソードを得る．得られたエピソードは E に加えて，新たに E'とする．最後に，E'中のエピソードに矛盾がないか調べ，矛盾があった場合には，E'からそれらのエピソードを除く．inconsistent-episodes はエピソードの集合を受け取り，互いに矛盾するエピソードと，矛盾があるエピソードを獲得するのに使われたエピソードを要素とする集合を返す．パターン獲得からパターンマッチによる新たなエピソード獲得までの処理は，エピソード獲得前のエピソードの集合を記憶しておき，パターンマッチ後のエピソードの集合と比較することで，新たなエピソードが得られなくなるまで繰り返される．

5. 評価

Web 上の表形式データに対して，4 章で述べたアルゴリズムを適用した．対象としたのは Web 上で入手できる PC 部品のカタログの Excel ファイルである．

一つの表に対し，一つまたは二つのエピソードを与えてアルゴリズムを適用した結果，表の規模に応じて数十から数百のエピソードが得られた．どのようなエピソードが得られるかは，初めにどのようなエピソードを与えるかに左右される．

例として表 5 からのエピソードの獲得を考える．表 5 では一つのインスタンスが一つの行で表され，各列にはインスタンスのプロパティ値が記述されている．一つの幅の広いセルから構成される行はインスタンスの分類を表す．またこの表では，

“NOTEBOOK COM-PUTERS”や“MOTHERBOARDS”のような大分類と“TARGA”や“TARGA ACCESSORIES”のような小分類の二階層の分類が用いられている.

表 5 階層的な分類を持つ表

NOTEBOOK COMPUTERS		
TARGA		
NOTTVXP	Targa Visionary XP AMD Mobile 2400 CPU	$ 2,099.00
NOTTXP-3	Targa Visionary XP-3 Intel P4 2.4 CPU	$ 2,359.00
NOTTS1	Targa Traveller S1 intel P4 2.4 CPU	$ 2,059.00
NOTTTC50DC	Targa Traveller C50 intel PENTIUM-M 1.3 CPU	$ 2,459.00
TARGA ACCESSORIES		
NOTTXP3DC	UPGRADE TO DVD WRITER FOR XP-3 ONLY	$ 269.00
NOTUPCPU5	UPGRADE TO INTEL P4 2.67Ghz CPU	$ 49.00
NOTUPCPU4	UPGRADE TO INTEL P4 2.8Ghz CPU	$ 69.00
NOTUPCPU3	UPGRADE TO INTEL P4 3.06 Ghz CPU	$ 449.00
IBM		
1829-4AM	TP PM-1.4GHz,256MB,30GB,14.1TFT,DVD	$ 2,239.00
1829-6DM	R50 PM-1.5GHz,256MB,40GB,15TFT,Combo	$ 2,745.00
1829-77M	TP PM-1.6GHz,512MB,60GB,15SXGA+,Combo	$ 3,569.00
1830-48M	TP PM-1.4GHz,256MB,30GB,14.1TFT,DVD	$ 2,635.00
1830-56M	TP PM-1.4GHz,256MB,30GB,14.1TFT,Combo	$ 2,780.00
MOTHERBOARDS		
AMD ATHLON THUNDERBIRD, DURON in SOCKET A FORM.		
MOB-8K9A2+	EPOX KT400,SOC-A,3DDR400,6PCI	$159.00
MOB-8K9A9I	EPOX KT400,SOC-A,3DDR400,5PCI	$125.00

この表に対して, 以下の 2 つのエピソードを与えてアルゴリズムを適用し, 表中のデータに関する新たなエピソードの獲得を行った.

・ クラス“TARGA”はクラス“NOTEBOOK COM-PUTERS”のサブクラスであり, そのインスタンスがプロパティ hasProductCode の値として“NOTTS1”を, プロパティ hasName の値として“Targa Traveller S1 intel P4 2.4 CPU”を, プロパティ hasPrice の値として”$2,059.00”を持つ

・ クラス IBM はクラス“NOTEBOOK COMPUTERS”のサブクラスであり, そのインスタンスがプロパティ hasProductCode の値として“1829-77M”を, プロパティ hasName の値として“TP PM-1.6GHz, 512MB,60GB,15SXGA+,Combo”を, プロパティ hasPrice の値として”$3,569.00”を持つ

アルゴリズム適用の結果, 表 5 から 1304 のエピソードが得られ, そのうち 1115 のエピソードが表の記述を正しく解釈したエピソードであった. 新たに獲得されたエピソードでは, ”TARGA ACCESSORIES” が ”NOTEBOOK COMPUTERS”のサブクラスであること, “AMD ATHLON THUNDERBIRD, DURON in SOCKET A FORM.”が“MOTHERBOARDS”のサブクラスであるなど, クラス階層に関する記述が得られた. また表に記述された多数のインスタンスについて, プロパティ hasProductCode, hasName, hasPrice の値が得られた. ここで得られたクラス階層は表中のインスタンスの分類の階層関係に沿ったものであり, 必ずしも一般的に用いることができるクラス階層であるとは言えない. しかし, ごく少ないエピソードからそれらのクラスに含まれる多数のインスタンスについてプロパティ値が得られており, 少ないコストで元の表のデータへのアノテーションが行えたと言える.

表の記述を誤って解釈して得られた 189 のエピソードでは, クラス階層が正しく得られていなかった. これは, 与えたエピソードからは“NOTEBOOK COMPUTERS”と“TARGA”のような大分類と小分類の二階層の分類に対応したパターンが得られるのにもかかわらず, 実際には一部のインスタンスは大分類しか持たないことが理由であった. 元の表では, 大分類と小分類はセルの背景色によって区別されており, 表構造の差異がないために, 誤ったエピソードが得られることになった.

このように異なる種類のデータが同じ表構造で記述されている場合には, 誤ったエピソードが得られることがある. また与えるエピソードが表の記述に沿わないものであった場合にも, 誤ったエピソードが得られる.

一般に同じ表構造でも表によってそのセマンティクスが異なるため, ある表で得られた表構造パターンは別の表で利用することはできない. しかしエピソードについてはある程度一般性があるため, ある表で得られたエピソードを使って他の表で表構造パターンの獲得を行うことができる.

ある表に以下のようなエピソードを与え, 新たなエピソード獲得のアルゴリズムを適用した.

・“Pentium 4 – 2.80GHz” はクラス “CPU”.のインスタンスである

その結果, クラス“CPU”とのクラス-インスタンス関係を記述する 32 のエピソードが得られた. さらにこれらのエピソードを用いて, Web から収集された約 2000 の Excel ファイルを対象にパターンの獲得とエピソードの獲得を行うと, そのうち 7 つの表から新たに 317 のエピソードが得られた. 得られたエピソードのうち 305 件が元の表で表現された正しい関係を記述しており, 12 件は元の表を誤って解釈したものであった.

このように複数の表を対象にする場合には, 同じ語でも表によって異なって解釈されることがあるという問題がある. 表 6 では, “Processor”,“Memory”,“Hard Drive”はプロパティ名であると解釈できる. しかし, “Pentium4 3.4GHz”がクラス“Processor”のインスタンスであるというエピソードを与えて表 6 から表構造パターンの獲得をおこなうと, 得られた表構造パターンを右にずらしてマッチさせた場合に, “256MB”や“128MB”がクラス“Memory”のインスタンスであるというエピソードが得られる. しかしこの表では“Memory”はプロパティとして解釈し, 厳密には“256MB”や“128MB”はデータタイプ“メモリサイズ”のインスタンスであると考えるのが正しい. これは, “Processor”という語だけでは, それがクラス名とプロパティ名のどちらにも解釈できるためである.

表 6 複数の解釈が可能な語を含む表

	Processor	Memory	Hard Drive
PC01	PentiumM1.7GHz	256MB	60GB
PC02	Pentium4 2.4GHz	256MB	120GB
PC03	Pentium4 3.4GHz	512MB	180GB

またリソースやプロパティが表に現れているかという判断は, 単純な文字列の一致(類似)に頼っているため, 同じものの表記に多くのバリエーションがある場合には, 獲得されたエピソードに記述されているリソースやプロパティを表中に発見できず, 表構造パターンが獲得できなくなる. この問題については表中の

データとエピソード中に記述されるリソースやプロパティとの一致を判定するのに，単語ごとの類似度を利用する方法[9]を用いることで，より多くの表から表構造パターンが得られると考えられる．

6. 結論

本研究では，広く用いられている表形式データから，オントロジーを自動的に構築する手法を提案した．本研究の貢献は以下の通りである．

- 表のデータ内容に依存しない表の解析
  提案手法では，隣接するセルの大小関係や同じ構造の繰り返しなどの表形式データの記述の特徴に基づく表構造の表現と，表構造の表す関係とを対応付けるため，表中のデータの内容によらず適用可能である．また，表構造の解釈に必要な前提知識はごく簡単なものでよいため，容易にさまざまなドメインに対して適用することができる．
- 表中のデータの複雑な関係の獲得
  提案手法では前もって与える表中のデータの関係の記述の記述に沿って表構造を解釈し，新たなデータ間の関係を得るため，属性名とその値のペアに限らず，クラスの階層関係・クラス-インスタンス関係・インスタンスとそのプロパティ値やそれらの組み合わせなど，さまざまな関係の記述が得られる．

さらに，提案手法の有用性を確認し，問題点を探るために，Web 上の表形式データに対して提案手法を適用した．その結果，表中のデータ間の関係についての簡単な記述を与えると，表中の数多くのデータについての関係の記述が高い精度で得られることがわかった．一方で同じ表の中でも同じ表構造が異なる意味を持つ場合があり，その場合には誤ったデータが得られた．対象とする関係を限定した場合には，得られた結果を用いて数多くの表を対象に処理を繰り返すことで，より多くのデータについての関係が得られることが確認できた．

今後の課題としては，Web から収集された大量の表形式データからオントロジーを構築し，それらを統合することが挙げられる．複数の表から得られたオントロジーを統合することにより，多くの表を対象に横断的な検索が可能になる．また多くの表から得られたデータを利用することで，獲得されたエピソードの正確さを検証したり，さらに多くのメタデータを自動的に生成することができると考えられる．

参考文献

[1] A.Maedche : Ontology Learning for the Semantic Web, Kluwer Academic Publishers (2002).

[2] N.Ashish and C.A.Knoblock : Wrapper Generation for Semi-Structured Internet Source, *ACM SIGMOD Records*, Vol.26, No.4, pp.8-15 (1997).

[3] S.Tsai H.Chen and J.Tsai : Mining tables from large scale HTML texts, *18th International. Conference. Computational Linguistics*, pp.166-172 (2000).

[4] M.Yoshida, K.Torisawa and J.Tsujii : Extracting ontologies from World Wide Web via HTML tables, *Pacific Association for Computational Linguistics*, pp.332-341 (2001).

[5] Y.A.Tijerino, David.W.Embley, D.W.Lonsdale and G.Nagy : Ontology Generation from Tables, 4th Interna-tional *Conference on Web Information Systems Engineering*, pp.242-252 (2003).

[6] A.Pivk, P.Cimiano and Y.Sure : From Tables to Frames, *3rd International Semantic Web Conference*, pp.166-181 (2004).

[7] M. Hurst : The Interpretation of Tables in Texts, PhD thesis, University of Edinburgh (2000).

[8] A.Amano and N.Asada : Complex Table Form Analysis Using Graph Grammar, *5th International Workshop on Document Analysis Systems*, pp.283-286 (2002).

[9] W. Cohen, P. Ravikumar and S.E. Fienberg : A Comparison of String Distance Metrics for Name-Matching Tasks, *IJCAI 2003 Workshop on Information Integration on the Web*, pp.73-78 (2003).