社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－DBS－134　(II)　(86)
2004／7／15

電子商取引における商品ルールの動的制約表現への変換手法の実装

志賀 隆之[†]　　小澤 正幸[‡]　　岩井原 瑞穂[†]

[1]商品ごとに多種多様な形式のルールが付され，それを適用することで価格や利用条件といった商品特性が変化するような性質を持つルール付商品では，ルールは通常，自然言語で記述されている．しかし電子商取引においては，ルールを用いた商品の検索や比較，検証を行う為，このようなルールは計算機による処理が行える形式でも表現されることが望ましい．このような変換処理を支援し，省力化などを実現する際には，表記の揺らぎの解決など様々な問題が存在する．本稿では商品ルールのモデル化を行ったうえで，日本語で記述された航空券のルールを例にとり，計算機での処理のための記述である動的制約表現への変換を，同一論理構造のルールのまとめ込みや，それへの変換テンプレートの適用などで支援する手法を提案し，その実装を示し，評価を行う．

Implementation for Converting Business-Rule to Dynamic Constraint Expression

Takayuki SHIGA[†]　　Masayuki KOZAWA[‡]　and　Mizuho IWAIHARA[†]

E-commerce is becoming popular, product rules, which describe various conditions such as prices, applicability, discount conditions are still expressed in natural languages. In the area of e-commerce, however, it is desirable to provide computer-processible rules for searching, comparing and validating merchandise. It is important to support of converting rules in a natural language into logical language for this purpose and realizing the laborsaving, but there are many problems such as solving the fracutuations among descriptions.　In this paper, we model business rules and take the case of airline tickets described in Japanese as an example.　Then, we propose a method for converting these rules into dynamic constraint representation for computer processing, we also show implementation of the process. This method groups rules whose logical structures are same and supports application of conversion templates for these groups.

1. はじめに

電子商取引の普及に伴い，従来は人手で行われていた分野の商取引活動に対しても計算機による支援が求められるようになってきている．筆者らはその中でもビジネスルールの自動処理に着目し，ルール付商品，つまり商品ごとに多種多様な形式のルールが付され，その適用によって価格などの商品属性が変化するという特徴を持つ商品の電子商取引に対して，そのルール処理の支援を行う仕組みを研究してきた[4], [5], [7], [8]．実際に電子商取引市場で扱われている商品の中で，このような特徴を持つルール付商品の具体例としては，航空券やホテル，レンタカー，旅行パックや PC の BTO 販売などが挙げられる．航空券では TravelXML や Open Travel Alliance Specification などの記述言語が提案されているが，いずれも論理的なルールの記述は自然言語により行なわれ，人間による解釈を前提としている．このような論理的表現は，将来的には商品データを作成する段階でデータ作成者によって与えられることが考えられる．しかし，現状では，Web ドキュメントのほとんどのルールデータが自然言語で記述されており，自然言語からこのような表現への変換作業が必要となる．こうした作業の省力化は，たとえば SemanticWeb[1] においても重要な問題である．RuleML[2]や BRML[3] といったルール記述のための XML 言語も提案されている．ウェブ文書からのビジ

[†]京都大学大学院情報学研究科
Department of Social Informatics Graduate School of Informatics, Kyoto University
[‡]凸版印刷株式会社
Toppan Corporation

ネスルール抽出では[6] が書店の送料ルールを例に Web からのルール抽出を行っているが，人手で自然言語のルールにマークアップする作業を必要としている．本論文では航空券販売を例に挙げ，筆者らが開発を行っているビジネスルール処理のフレームワークである動的制約データベース用の表現[7]に変換する作業を支援する手法を議論する．これは他の論理的表現への変換についても,共通して適用できうる部分を有する．

本論文の構成は，まず 2.節でルール付商品とその例である航空券について述べ，3.節でルール付商品のモデル化を行う．そして，4.節では自然言語で記述されたルールの論理的表現への変換について述べ，5.節ではその変換支援手法のアルゴリズムと実行例を示す．6.節で実装の概要と結果，その考察を行い，7.節で結論を述べている．

2. ルール付商品

現在，国内でも旅行代理店・航空会社などが運営するサイトで電子商取引として航空券が販売されている．その取引形態は様々であり，Web サイトのみで商品選択と購入が可能なカタログ販売型とサイト上には簡潔な情報のみを置き，詳細については電子メールなどによって顧客と旅行代理店間で交渉を行う相対取引型が一般的である．一般に格安航空券と呼ばれるものは，航空会社からまとめて仕入れた商品に，様々な条件を付けて小売りするものであり，航空会社のものより価格は安いが様々な制約条件が付帯されているものが多い．この場合は様々な割引方式や複雑な価格構成が存在し，ルールには商品ごとのばらつきが見られる．ただし，商品ルールの自然言語記述には，表現の揺らぎはあるが，文の意味自体は曖昧性を持たないという特徴がある．実際の航空券から抜き出したルールの例を表 1 に示す．

訪問可能都市	アトランタ，ボストン，バッファロー，…，シアトル，サンフランシスコ，ソルトレイクシティ，バンクーバー，トロント
訪問可能都市数	●上記指定都市内 3 都市周遊可能です．
有効期限	1 泊 3 日～1 ヶ月以内 FIX（復路変更不可）
追加料金	週末料金：現地発金・土・日は，¥4,000UP
ストップオーバー	2 回まで無料．3 回目まで可：¥10,000UP

表 1: 航空券の商品ルールの例

こういった商品ルールを本稿で扱うような論理的表現に変換することにより，以下のような計算機による有益な応用が考えられる．

- 売り手の条件と買い手側の条件をそれぞれルールで表現し，それらの論理積の充足可能性や，含意の関係を調べることにより，互いの条件が満足されているかの検査を行う．
- 商品にどのような付帯条件や制限事項があるかを検索し，表示する検索機能．
- 買い手の要求条件を，商品ルールに代入することにより，可能な選択肢を求め，それを買い手に選択メニューとして提示する．

3. 商品ルールのモデル化

3.1. 論理的表現の要求事項

ルールの表現手法に関して望ましい性質としては次のようなものが挙げられる．

- 論理的表現には曖昧性が無く，厳密な意味が与えられることが必要である．
- 計算処理が実用的時間になるように，表現力を適切なものとする．
- 論理的表現から元の自然言語の表現が直ちに求められるようにする．
- 実際には自然言語で記述されているルールが表す概念の全てが論理的表現に変換できるとは限らないため，部分的に自然言語表現を混在させることが必要である．自然言語による表現の部分は，利用者に表示して真偽値の判断を求めるようにする．

3.2. 商品ルールの論理的表現のモデル

定義 1

1. 商品ルールは商品の各種特性を表す商品属性とその値を表す商品属性値からなる制約を表わす論理式である．
2. 1 つの商品ルールは，いくつかのルール条項に分割することができ，各商品ルールはルール条項の論理積である．
3. 各商品属性には値域集合が割り当てられている．これはたとえば日付や金額，地名等である．
4. 商品ルールあるいはルール条項の自然言語による表現をそれぞれ商品ルール文およびルール条項文と呼ぶ．
5. 商品ルールおよびルール条項を表わす論理式のクラスとして，等号論理を用いる．これは，変数と定数，または変数同志の等号を *AND*,*OR*,*NOT* のブール結合子で結合した式であり，例えば $(x = "abc" \vee y \neq "def") \wedge \neg (z = "ghi")$ のような論理式である．各変数はいずれかの商品属性に対応付けられ，定数は商品属性の値域集合に含まれるものとする．

上記の等号論理式の制約集合に対し，制約データベース[9]の考え方により代数演算を行う質問言語として，動的制約代数[4][5] があり，商品ルールの集合的操作および論理演算に用いることができる．

4. ルール条項文の変換処理

4.1. 変換元の航空券データ

変換元のデータとして，旅行代理店による航空券販売サイトであるアルキカタ・ドット・コム(http://www.arukikata.com/) から 3 ヶ月分 2000 件の航空券データを収集した．データには商品コードや訪問可能都市，出発地，出発日と基本料金の対応表，追加料金ルール，その他のコメントなど合わせて計 17 の項目分けがなされ，日本語で記述されている．上記の日本語の記述

は商品ルール文に対応することになる．商品ルール文をさらにルール条項文に分割してゆく必要があるが，意味的にまとまったものを分割するために，改行文字や箇条書きの区切りなどを手がかりとし，スクリプト処理を行った[8]．この分割では他を参照しなければ意味を把握できないルール条項文を作らないように，分割の基準を緩やかに取った．この作業により，収集した 2000 件の商品データに付随する商品ルールを分割したルール条項の数は 18482 件となった．さらに文字列の比較により重複したものを削除すると 3766 種類である．

4.2. 類似ルールの例

商品ルールの中には，他と同一か類似するルール条項文は多く，また日本語の表記の揺らぎがあるものや，論理的な構造は同じでも，具体的な値やその個数が異なるものがある．収集したルールのうち，週末料金に関するルール条項文の中から，このような表記の揺らぎや具体的な値，そして値の個数の違いを含む例を表 2 に示す．

1	*帰路ソウル発が 10/14・11/4・3/23 の場合、¥26,000UP となります.
2	*復路、クアラルンプール発が 10/13・11/3・12/22・1/12・3/23 は¥7,000UP.
3	*復路、ソウル発 10/14・11/4・3/23 は¥26,000UP
4	*復路、マニラ発が 10/14・11/4・12/23・1/13・2/11・3/23 の場合は、¥5,000UP.（週末料金別途必要）
5	●復路：グアム発 12/23、1/04-1/05、1/13、2/11、3/23 は、¥6,000UP
6	●復路：シンガポール発 1/04、1/05 は¥12,000UP
7	●復路：ソウル発：10/14、11/04、1/04、1/05、1/13、3/23 は¥10,000UP
8	●復路：ソウル発：12/23、2/11 は¥8,000UP
9	●復路：ホノルル発 1/02、1/03 は、¥15,000UP
10	●復路：ホノルル発 11/03、12/09-12/11、12/22、1/05、1/12、3/22 は、¥5,000UP
11	●復路：香港発：1/04-1/05、1/13、2/11、3/23 は、¥6,000UP
12	●復路：香港発：10/14、11/04、1/04-1/05、1/13、2/11、3/23 は、¥6,000UP
13	復路：シンガポール発 1/04-1/05 は、¥10,000UP
14	復路：シンガポール発 1/04、1/05 は¥10,000UP
15	復路：シンガポール発 1/04、1/05 は、¥10,000UP
16	復路：マレーシア発 1/04、1/05 の場合¥10,000UP.
17	復路：マレーシア発 1/04、1/05 は¥10,000UP
18	復路：マレーシア発 1/04、1/05 は、¥10,000UP
19	復路：マレーシア発　1/04、1/05 の場合は、¥10,000UP
20	復路：マレーシア発 1/04・1/05 は、¥10,000UP
21	●復路、ソウル発 10/14・11/4・3/23 は¥26,000UP
22	●復路、ソウル発 10/14、11/4、3/23 は¥26,000UP
23	●復路、ソウル発 10/14・11/4・3/23 の場合¥26,000UP
24	●復路：ソウル発 10/14・11/04・3/23 は¥26,000UP
25	復路　ソウル発 10/14・11/4・3/23 は¥26,000UP
26	復路：ソウル発 10/14・11/4・3/23 は¥26,000UP
27	復路：ソウル発 10/14・11/4・3/23 の場合¥26,000UP
28	復路：シンガポール発 1/04、1/05 の場合、¥10,000UP
29	復路　シンガポール発 1/4・1/5 では¥10,000UP
30	復路：シンガポール発 1/04，1/05 は¥10,000UP

表 2：表記の揺らぎなどを含むルール条項文

- 行頭の“*”や“●”は箇条書きの先頭記号に由来する．統一されたフォーマットが存在するわけではなく，ルール条項文の記述者によると見られる不揃いが存在する．
- ルール 1,3,21,22,23,24,25,26,27 は表記の違いだけで，内容としては「復路にソウルを出発する日時が 10/14 または 11/4 または 3/23 であれば，26000 円の追加料金がかかる」という同一のものである．
- 追加料金の対象となる日時の値の個数は 2 日から 6 日まで様々である．
- ルール 6 と 13 では，都市または日時と追加料金額の対応関係が 2 通り示され，他の物と異なる．

5. 商品ルールの論理的表現の抽出支援手法

5.1. 登録単語辞書

商品ルール文を解析し，論理的表現に変換するために，商品ルール文の記述に使用されている用語が登録されている登録単語辞書を用意する．登録単語のカテゴリーを表す値域集合の名前を単語型あるいは単に型と呼ぶ．たとえば「都市名」や「日時」，「曜日」，「金額」，「否定語」が単語型である．登録単語辞書では，登録語から単語型を引けるようにする．また，商品ルール文は人手で書かれたものが多いため表記の揺らぎが存在する．例えば，土曜日を表す語としては，“土”，“土曜”，“土曜日”などがあり，このうちの 1 つを標準形として辞書に登録する．

5.2. 論理的表現の抽出支援手法の流れ

1. 商品ルール文について，登録単語辞書を検索して標準形の登録語(トークン)に置換する．この過程をトークン抽出と呼ぶ．トークン抽出により，表記の揺らぎが吸収される．
2. 航空券においては，訪問可能都市や可能な日時・曜日の列挙が見られるが，これらは同じ型のトークンの列として検出可能である．トークンの反復畳み込みとして，このようなトークンの列を検出する．これにより，トークン列の繰り返し回数や，カンマ等の句読点を無視した，同じ型のトークン列を認識する．
3. ルールのクラスタリングとして，トークンの反復畳み込みの結果が同一になったルール条項をまとめ込む．
4. 変換テンプレートの適用として，上記の結果のル

ール条項文にマッチする変換テンプレートを検索し，そのテンプレートに従ってルール条項文を論理的表現に変換する．

上記の手法は，自然言語という曖昧な言語を処理するという特性上，完全に自動化できるものではなく，変換に誤りが含まれることがある．そのため，各段階で人手によって変換結果の確認を行う．最終的な判断を人間が行うという意味で，本手法は抽出支援手法という位置づけになる．数千を越える商品ルールを全て人手で行うのは困難であり，本抽出支援手法による半自動化の効果は大きいと考えられる．

5.3. 抽出支援手法の実行例

5.3.1. ルールからのトークン抽出

ルールからトークン抽出を行う際，揺らぎのある表記は標準形であるトークンに変換される．表 2 のルール 1,13,24,29 に対しトークン抽出を行った結果を，表 3 に示す. 型名の先頭には#を, 標準形には下線を付し，ルール文中に現れた表記については，二重引用符でかこった．これらについて型と標準形のみで見た場合，ルール 1 と 25, そして前述の通りこれと同じ意味内容を持つルール(例えば 3,21,22,23,25,26,27)は，全て一致し，表記の揺らぎを吸収できたことが分かる．

	(#型, 標準形,“ルール中の表記”)
1	(#復路, 復路, “帰路”) (#都市, ソウル, “ソウル”) (#日時, 10/14, “10/14”) (#日時, 11/4, “11/4”) (#日時, 3/23, “3/23”) (#追加金額, ¥26000, “¥26,000UP”)
13	(#復路, 復路, “復路：”) (#都市, シンガポール, “シンガポール”) (#日時, 1/4, “1/04”) (#日時, 1/5, “1/05”) (#追加金額, ¥10000, “¥10,000UP”)
24	(#復路, 復路, “●復路：”) (#都市, ソウル, “ソウル”) (#日時, 10/14, “10/14”) (#日時, 11/4, “11/04”) (#日時, 3/23, “3/23”) (#追加金額, ¥26000, “¥26,000UP”)
29	(#復路, 復路, “復路 ”) (#都市, シンガポール, “シンガポール”) (#日時, 1/4, “1/4”) (#日時, 1/5, “1/5”) (#追加金額, ¥10000, “¥10,000UP”)

表 3：トークン抽出を行ったルール条項文

5.3.2. トークンの反復畳み込みとクラスタリング

トークンの反復畳み込みについて述べる．トークン列 $S = s_1 s_2 ... s_l$ について，各トークン s_i の型 t_i を調べ，型の部分列 $t_j, t_{j+1}, ..., t_{j+k}$ が 1 回以上の繰り返しになっている箇所が見つかれば，それを型の列の繰り返しを表わすトークン $(t_j, t_{j+1}, ..., t_{j+k})^+$ で置換する．この操作を S の先頭から順次行い，適用できなくなるまで変換する．畳み込まれたトークン $(t_j, t_{j+1}, ..., t_{j+k})^+$ を縮退トークンと呼ぶ．縮退トークン列の集合について，縮対トークン列の型が一致するものをまとめ込む操作をルールのクラスタリングと呼ぶ．そのクラスタに対応するルール条項文の集合は, 表記の揺らぎや具体的な値の違い, そして反復回数の違いを無視した上で同じ構造を持つルール条項文をまとめた集合となる．

表 3 の例に対して，反復畳み込みを行った結果を表 4 に示す．ルール 1,24 は a であらわされる列型に，ルール 13,29 は b で表される列型を持つ縮退トークン列に畳み込まれる．この他にも，表 2 で示した例のいくつかがこれら a, b いずれかの列型を持つ縮退トークン列に畳み込まれる．

	列型
a	“復路”[1]; “ソウル発”[2]$(_1$日時[3]$)_1{}^+$ 金額[4]
b	“復路”[1]; “シンガポール発”[2]$(_1$日時[3]$)_1{}^+$ 金額[4]

表 4：まとめ込まれたトークン列の型

5.3.3. 変換テンプレートの適用

得られた縮退トークン列の集合に対し，縮退トークン列ごとに論理的表現への変換を与える変換テンプレートを用意する．変換テンプレートでは，縮退トークンの型および定数に応じて，変数および定数を割り当て，さらに縮退トークンの括弧の入れ子構造に対応して，論理演算を割り当ててゆく情報を持たせる．

縮退トークン列 τ の各部分列に変換テンプレートの対応する変換を適用するために，τ の開き括弧と閉じ括弧に同じ番号からなる識別子を与え，また τ に f 含まれる各トークン型にも先頭から番号の識別子を与える．表 4 には，型に番号付けを行っており，トークン型の番号は上付で, 括弧の番号は下付で示されている．

次に論理的表現への変換する際の“～は不可”といった否定を含む表現について考える．すべての否定演算子“¬”はド・モルガンの法則によって，$v = "c"$ または $v \neq "c"$ といったリテラルに押し下げることができる．これにより，商品ルール文の大部分は，変換結果として AND(∧)と OR(∨), または NOR(⊕)で表記できる．そして，括弧に対応する部分列に対する変換規則について考える．各関数の計算は文字列操作などであり，変換スクリプトとして定義される．変換関数の定義において，“.”は文字列連接の演算子であり，$std(c)$ は c に対するトークンの標準形を与える．例えば，型が都市であるトークン列 $c_1, c_2, ..., c_k$ は $(_i w_j)_i^+$ と表現される．$\tau((.)_k) = \vee$，$\tau(w_j) = city1$ とすると，ルール文は次のように変換される．

$$(city1 = std(c_1) \vee city1 = std(c_2) \vee \cdots \vee city1 = std(c_k))$$

また，論理演算子として $\tau((.),)$ に NOR(⊕)を用いると，このルールは次のように表せる．

$(city1 \neq std(c_1) \wedge city1 \neq std(c_2) \wedge \cdots \wedge city1 \neq std(c_k))$

動的制約代数はこのような変数と定数による等式とその否定を OR で結んだ和項，あるいはその否定を AND で結んだ積項を扱えることが特徴である．

次に，変換テンプレートの定義を行う．列型 τ を持つ縮退トークン列集合 C_τ への変換テンプレート MC_τ とは，以下の規則に従う，τ の各構成要素への，変換関数 f，論理演算子 op，空文字 ε のいずれかの割り当てである．構成要素 a に A を割り当てることを $MC_\tau : a \mapsto A$ と書く．変換テンプレートの形式的定義は省略するが，以下の適用例から容易に理解できるものと考える．表 5 にこの変換テンプレートの例を示す．

MC_τ ：$(.)_1 \mapsto$ '∨'
MC_τ ：$(.)_1$ #金額 $^4 \mapsto$ '∧'
MC_τ ：#都市 $^2 \mapsto f_2$
MC_τ ：#日時 $^3 \mapsto f_3$
MC_τ ：#金額 $^4 \mapsto f_4$

表 5：変換テンプレートの例

また，変換テンプレートの種類としては，意味型変換テンプレート(semantic transformation)と包括型変換テンプレート(generic transformation)を定義する．意味型変換テンプレート τ_s では，登場した文字列 $<t_1,t_2,...,t_k>$ について，意味に応じて，変数や定数，論理演算子を逐次割り当てることができる．一方，包括型変換テンプレート τ_{gen} は，登場したすべての文字列に変数と定数を自動的に割り当てるというものである．意味型変換テンプレートは様々な文字列パターンに対し，正確に変換を行うことができるが，テンプレートを一つ一つ用意しなければならず，コストがかかるという欠点がある．一方，包括型変換テンプレートでは各々の文字列の意味に応じたテンプレートを適用する必要がなく，コストが少ないという長所がある反面，逆に，ルール文のすべての意味を論理式に反映できないという短所もある．結局，理想的な変換テンプレートとは，各々のルールに対して，必要に応じてユーザと対話しながら精度を向上できるものであると考えられる．

表 2 のルールに対して，包括型変換テンプレートを適用した結果を表 6 に，意味型変換テンプレートを適用した結果を表 7 に示す．表 6 においては，“登場した文字列の型名＋登場順”で変数名を構成している．一方，表 7 における変数，“CityDepCome”は復路の出発都市を，“DateDep”は出発日時を，“PriceUp”は追加料金を表している．

●City1=“クアラルンプール” ∧ Date1=“10/13” ∧ Date2=“11/3” ∧ Date3=“12/22” ∧ Date4=“1/12” ∧ Date5=“3/23” ∧ PriceUp1=“7000”
●City1=“グアム” ∧ Date1=“12/23” ∧ Date2=“1/04-1/05” ∧ Date3=“1/13” ∧ Date4=“2/11” ∧ Date5=“3/23” ∧ PriceUp1=“6000”
●City1=“シンガポール” ∧ Date1=“1/04” ∧ Date2=“1/05” ∧ PriceUp1=“10000”
●City1=“シンガポール” ∧ Date1=“1/04” ∧ Date2=“1/05” ∧ PriceUp1=“12000”
●City1=“シンガポール” ∧ Date1=“1/04-1/05” ∧ PriceUp1=“10000”
●City1=“ソウル” ∧ Date1=“10/14” ∧ Date2=“11/04” ∧ Date3=“1/04” ∧ Date4=“1/05” ∧ Date5=“1/13” ∧ Date6=“3/23” ∧ PriceUp1=“10000”
●City1=“ソウル” ∧ Date1=“10/14” ∧ Date2=“11/4” ∧ Date3=“3/23” ∧ PriceUp1=“26000”
●City1=“ソウル” ∧ Date1=“12/23” ∧ Date2=“2/11” ∧ PriceUp1=“8000”
●City1=“ホノルル” ∧ Date1=“1/02” ∧ Date2=“1/03” ∧ PriceUp1=“15000”
●City1=“ホノルル” ∧ Date1=“11/03” ∧ Date2=“12/09-12/11” ∧ Date3=“12/22” ∧ Date4=“1/05” ∧ Date5=“1/12” ∧ Date6=“3/22” ∧ PriceUp1=“5000”
●City1=“マニラ” ∧ Date1=“10/14” ∧ Date2=“11/4” ∧ Date3=“12/23” ∧ Date4=“1/13” ∧ Date5=“2/11” ∧ Date6=“3/23” ∧ PriceUp1=“5000”
●City1=“マレーシア” ∧ City2=“マレ” ∧ Date1=“1/04” ∧ Date2=“1/05” ∧ PriceUp1=“10000” ∧
●City1=“香港” ∧ Date1=“1/04-1/05” ∧ Date2=“1/13” ∧ Date3=“2/11” ∧ Date4=“3/23” ∧ PriceUp1=“6000”
●City1=“香港” ∧ Date1=“10/14” ∧ Date2=1“1/04” ∧ Date3=“1/04-1/05” ∧ Date4=“1/13” ∧ Date5=“2/11” ∧ Date6=“3/23” ∧ PriceUp1=“6000”

表 6：包括型変換テンプレートの適用例

●CityDepCome=“クアラルンプール” ∧ (DateDep=“10/13” ∨ DateDep=“11/3” ∨ DateDep=“12/22” ∨ DateDep=“1/12” ∨ DateDep=“3/23”) ∧ PriceUp=“7000”
●CityDepCome=“グアム” ∧ (DateDep=“12/23” ∨ DateDep=“1/04-1/05” ∨ DateDep=“1/13” ∨ DateDep=“2/11” ∨ DateDep=“3/23”) ∧ PriceUp=“6000”
●CityDepCome=“シンガポール” ∧ (DateDep=“1/04” ∨ DateDep=“1/05”) ∧ PriceUp=“10000”
●CityDepCome=“シンガポール” ∧ (DateDep=“1/04” ∨ DateDep=“1/05”) ∧ PriceUp=“12000”
●CityDepCome=“シンガポール” ∧ (DateDep=“1/04-1/05”) ∧ PriceUp=“10000”
●CityDepCome=“ソウル” ∧ (DateDep=“10/14” ∨ DateDep=“11/04” ∨ DateDep=“1/04” ∨ DateDep=“1/05” ∨ DateDep=“1/13” ∨ DateDep=3/23”) ∧ PriceUp=“10000”
●CityDepCome=“ソウル” ∧ (DateDep=“10/14” ∨ DateDep=“11/4” ∨ DateDep=“3/23”) ∧ PriceUp=“26000”
●CityDepCome=“ソウル” ∧ (DateDep=“12/23” ∨ DateDep=“2/11”) ∧ PriceUp=“8000”
●CityDepCome=“ホノルル” ∧ (DateDep=“1/02” ∨ DateDep=“1/03”) ∧ PriceUp=“15000”
●CityDepCome=“ホノルル” ∧ (DateDep=“11/03” ∨ DateDep=“12/09-12/11” ∨ DateDep=“12/22” ∨ DateDep=

“1/05” ∨ DateDep=“1/12” ∨ DateDep=“3/22”) ∧ PriceUp=“5000”
●CityDepCome=“マニラ” ∧ (DateDep=“10/14” ∨ DateDep=“11/4” ∨ DateDep=“12/23” ∨ DateDep=“1/13” ∨ DateDep=“2/11” ∨ DateDep=“3/23”) ∧ PriceUp=“5000”
●CityDepCome=“マレーシア” ∧ (DateDep=“1/04” ∨ DateDep=“1/05”) ∧ PriceUp=“10000”
●CityDepCome=“香港” ∧ (DateDep=“1/04-1/05” ∨ DateDep=“1/13” ∨ DateDep=“2/11” ∨ DateDep=“3/23”) ∧ PriceUp=“6000”
●CityDepCome=“香港” ∧ (DateDep=“10/14” ∨ DateDep=“11/04” ∨ DateDep=“1/04-1/05” ∨ DateDep=“1/13” ∨ DateDep=“2/11” ∨ DateDep=“3/23”) ∧ PriceUp=“6000”

表 7：意味型変換テンプレートの適用例

これらの例を見てもわかるように,意味型では,個々のルールの意味を解釈して論理式化しているため，包括型と意味型を比べたときに変数名や AND，OR の処理面で，意味型のほうがより緻密に論理式として表せることがわかる．

6. 実装と評価

6.1. 結果

5.節で述べた手法に基づいて，辞書を元にルール条項文をトークン抽出し，反復畳み込みとクラスタリングを行い，その後，変換テンプレートを適用するというプログラムを Perl で実装した．変換テンプレートに関しては，前述の包括型と意味型の 2 通りの手法を適用した.前述の航空券データより抽出した 3766 件のルール条項文について,クラスタリングの結果 327 種(括弧付けのみが異なる物を含めると 493 種）のパターンに類別することが出来，包括型変換テンプレートを適用することで,これらをすべて処理することができた．意味型変換テンプレートでは，このうち 73 種（括弧付けのみが異なるものを含めると 117 種）まで処理することを可能にした．表 8 はクラスタリングにかかった計算時間，表 9 は各々の変換テンプレートの適用にかかった計算時間を示している．

	Parsing	Reducing	Grouping	Total
計算時間	9745	3	4	9752

表 8：クラスタリングにかかった計算時間（単位：秒）

	包括型	意味型
全体の計算時間	331	1600
クラスタ 1 つあたりにかかった計算時間	0.671	13.675

表 9：変換テンプレートの適用にかかった計算時間（単位：秒）

6.2. 評価

クラスタリングまでの処理に関して，単語の型として曜日，価格，日付，都市名など 11 種類の辞書データを用意し，それぞれに属する語をルール文から抜き出し登録することで辞書を作成した．辞書が完全でない段階では，変換誤りが見られたが，辞書に用語を追加することによりトークン抽出およびクラスタリングの精度の向上が見られた．

変換テンプレートの処理では，前章で述べたように意味型の方が包括型と比べて，より緻密にルールを論理式化できる反面，表 9 が示すように，計算時間がかかるという欠点がある．

また，現在のシステムでは変換した結果の誤りを人間が修正する形式を取っているが，登録単語辞書や変換テンプレートの修正過程で適宜入力して精度向上を図れるような対話型のシステムも今後の課題である．

7. まとめ

本論文では，Web 上に自然言語で記述されたルール付商品の商品ルールを論理的表現に変換するために，トークン抽出や変換テンプレートを用いた変換支援手法を提案し，それを実装することで評価することに成功した．今後は，さらなる手法の改良や計算時間の短縮を目指す予定である．

文　献

[1] Berners-Lee, T.: The Semantic Web-LCS seminar, http://www.w3.org/2002/Talks/09-lcs-sweb-tbl/

[2] Boley, H.: The Rule Markup Language: RDF-XML Data Model, XML Schema Hierarchy, and XSL transformations, INAP2001, Tokyo, Japan, in October 2001(2001).

[3] Grosof, B. N. , Labrou, Y. and Chan, H. Y.: A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML, E-COMMERCE 99, Denver, Colorado(1999).

[4] Iwaihara, M.: Supporting Dynamic Constraints for Commerce Negotiations, 2nd Int. Workshop in Advanced Issues of E-commerce and Web-Information Systems (WECWIS), IEEE Press, pp.12-20, June (2000).

[5] Iwaihara, M.: Matching and Deriving Dynamic Constraints for E-Commerce Negotiations, Workshop on Technologies for EServices, (informal proceedings), Cairo, Sep. (2000).

[6] Kang, Juyoung and Lee, Jae Kyu: Extraction of Structured Rules from Web Pages and Maintenance of Mutual Consistency: XRML Approach, Proc. of RuleML 2003, Springer-Verlag, LNCS2876, pp. 150-163, (2003).

[7] Kozawa, M. , Iwaihara, M. and Kambayashi, Y.: Constraint Search for Comparing Multiple- Incentive Merchandises, Proc. of EC-Web 2002, Springer, LNCS2455, pp.152-161, Sep. (2002)

[8] 小澤正幸, 岩井原瑞穂, 上林彌彦:電子商取引における商品ルール抽出支援手法, 第 15 回データ工学ワークショップ(DEWS2004)論文集 ISSN 1347-4413.

[9] Kuper, G. , Libkin L. and Paredaens, J. (Eds.): Constraint Databases, Springer-Verlog, (2000).