文型パターンパーサの試作

徳久 雅人　　村上 仁一　　池原 悟
鳥取大学 工学部 知能情報工学科
{tokuhisa, murakami, ikehara}@ike.tottori-u.ac.jp

1. はじめに

言語表現には，表現の意味が表現構造と独立に扱うことができないという非線形性の問題がある．等価的類推思考の原理に基づく機械翻訳方式[1]では，翻訳対象となる両言語で言語表現を「文型パターン」によって記述しておき，意味的に等価な文型パターンを対応付けることで，意味の失われない解析・生成を実現しようとしている．

文型パターンの記述は，網羅性と可読性を意識して設計されており，字面，変数，関数，記号が用いられている[2]．現在，日本語の重文・複文を対象とした文型パターンを大量に蓄積した「文型パターン辞書」の構築が進められており[3]，文型パターンの照合方式の設計とその実装が求められている．

そこで，本稿では，本翻訳方式の実現に向けて，文型パターン記述を受理し，全ての適合結果を出力する文型パターンパーサの試作を目的とする．まず 2.章で文型パターンの記述形式を示し，3.章で照合アルゴリズムを設計する．4.章では高速化について述べ，5.章で実装と実験について述べる．

2. 文型パターン

2.1. 文型パターンの記述形式

文型パターンは，日英文対応の対訳コーパスから作成する．原文(L)は，単語レベル(W)，句レベル(P)，節レベル(C)の 3 レベルにパターン化する．各レベルに応じた粒度でアライメントがとれた部分は，線形要素として変数化する．逆に，変数化すると対訳の訳出が困難になる部分は変数化せず，非線形要素として字面，あるいは，関数の形式で残す．具体例を示す．

LJ108660:彼は日本一のバイオリニストだと言われている。
LE071983:People say that he is the best violinist in Japan.
WJ096082:*N1*は/日本一の/*N2.da*と/*V3.reru.teiru*。
WE107379:People *V3* that *N1* be the best *N2* in Japan.
PJ000375:*N1*は/*NP2.da*と/*V3.reru.teiru*。
PE098545:People *V3* that *N1* be *NP2*.
CJ000043:*CL1.da*と/*V2.reru.teiru*。
CE001085:People *V2* that *CL1*.

たとえば，「日本一」の対訳は「the best ～ in Japan」に対応するため，単語レベルでは変数化しないが，句レベルになると「バイオリニスト」を含めて変数化する．

パターンの記述要素には，字面，変数，および，関数がある．記号は記述要素を制御する．日本語パターンの記述要素は入力文の解析のために働き，英語パターンの記述要素は英文の生成のために働く[2]．以降では，日本語パターンによる解析について概説する．

変数には，名詞や動詞の単語を表す *Nn* や *Vn* ， 名詞句や節を表す *NPn* や *CLn* などがある(*n* は識別番号)．変数は，入力文に適合すると，対応する入力文の部分の言語情報が代入(バインド)される．関数は，変数のバインド値の形式や字面の指定，あるいは，表現の統括を行う(表1)．記号は，パターン記述要素の適合の仕方について，任意化，選択，順序変更，記憶という制御を行う(表2)．

表1: 関数の一覧

種類	意 味 と 例
様相関数	変数に後続する表現の形式を指定 例 *V1.reru*
語尾関数	変数に対応する表現の形式を指定 例 *V1^katei*
字面関数	変数に含まれる字面を指定 例 #大変(*CL1*)
マクロ関数	引数内表現を指定する変数が統括 例 *#CL1*(*N2* が *V3*)

表2: 記号の一覧

記号名	表 記	意 味
離散記号	/	文型に無関係な要素
選択記号	(…|…)	いずれかの要素列と適合
任意記号	[…]	文型選択上，任意の要素
補完要素記号	<…>	ゼロ代名詞等
順序任意要素指定記号	{…,…}	順序入れ替え可能な範囲 (例 格要素の順序)
位置変更可能要素指定記号	$*n*^{…定義…} $*n*	指定位置に入れ替え可能 (例 副詞の位置)
文節境界記号	!	文節境界と適合
記憶記号	*#n*	適合内容を記憶

2.2. 文型パターン辞書の規模

既に 128,713 個の日英対訳文対から 3 レベルのパターンセットが上述の形式で作成されている．日本語パターンの数は，単語レベルは 124,158 パターン，句レベルは 87,685 パターン，そして，節レベルは 11,280 パターンである．

パターンで使われた変数の種類数から，表現構造の複雑さがわかる．単語レベルでは名詞，時詞，数詞，動詞，連体詞，副詞，形容詞，形容動詞の 8 種類，句レベルでは，動詞句，名詞句，形容詞句，形容動詞句，副詞句を加えて 13 種類，節レベルでは節を加えて 14 種類である．

照合の概念(3.章)によると，変数の定義で他の変数を用いる深さは2，3段階程度である．したがって，文型パターン辞書は，浅くて広い階層構造となる．

3. 文型パターンの照合

3.1. 文型パターン照合の概念

文型パターンの照合とは，文型パターン辞書から，入力文に適合する文型パターンを全て検索すると同時に，パターン記述要素の適合の仕方，すなわち，「適合解」を全て求めることである．文型パターンが入力文に適合する条件は，文型パターンの全ての必須の記述要素が，パターンの指定する順序通りに，入力文と対応することである．対応関係に曖昧性があるならば，全てを別々の適合解とする．

ここで，対応関係の曖昧性は2通りがある．1つは，対応する入力文の部分が異なる場合，もう1つは，対応部分の解釈が異なる場合である．たとえば，「太郎の弟の発言」という入力と名詞句 *NP* の対応をとる際，前者は *NP*=「太郎」，「太郎の弟」，

「弟の発言」など対応範囲が異なる場合が該当する．後者は *NP*=「太郎の弟の発言」であっても「太郎」が「弟の発言」にかかる解釈と「太郎の弟」が「発言」にかかる解釈という係り先の異なる場合が該当する．

前者の曖昧な対応関係は別々の適合解となる．後者の関係は，文型パターンが変数の解釈に制約を与えているならば別々の適合解となるが，文型パターンから制約のない変数ならば変数内の解釈は線形要素内の問題であるため，文型パターンにおける別解とはならない．

3.2. 文型パターン照合の概要

文型照合をする文は，形態素情報の形式で入力される．文型パターンの記述要素に文節境界を含むので，形態素境界を単位に入力文を参照する．形態素情報は品詞コード，標準表記，意味属性コードから成る．以下に具体例を示す．

入力文： ₁**今晩**₂**は**₃**もう**₄**かえれ**₅**。**₆

形態素： 1= /**今晩**(1500)， 2=+**は**(7530)， 3=/**もう**(4100)，
4=/かえれ(2189,**返る**,**返れ**,{NY:20,NY:30･･}) (2189,**帰る**,**帰れ**,{NY:18,･･})，
5=+。，6=/nil

ここで，「/」は文節境界を表し，「+」は非文節境界を表す．

以下の3パターンとこの例文を照合した結果を説明する．

PTN1： *TIME1*は/もう*V2^meirei*。

PTN2： *AJ1*ので/*TIME2*は/*V3^meirei*。

PTN3： *TIME1*は/*V2^meirei*。

PTN1は，過不足なく適合し，各変数は *TIME1*=「今晩」，*V2*=「かえれ」がバインドされる．PTN2は，「 *AJ1* ので」が入力文と対応しないため，不適合となる．PTN3は，「もう」が対応しないが，パターンの記述要素が全て対応するので，適合となる．

文型パターンの記述に記号による制御や関数によるバインド値の操作という機能を含むので，照合方式は，これらの記述が容易な ATN(Augmented Transition Network)[4]をベースとする．オートマトンのノードは解析状態に，アークは主に文型パターンの記述要素にそれぞれ対応する．原則として状態遷移は，形態素単位で行う．

文型照合は，常に全ての適合解を求めるので，適合解の管理の簡便化を狙い，エージェントの概念を用いた幅優先探索を行う．そこで，本照合方式を AB-ATN （Agent oriented Breadth-first ATN)と呼ぶ．

文型パターンの記述能力を発揮するために ATN では，アークの設計が重要になる．そこで，次節では，オートマトンの形成方法を示し，その後の節で，照合アルゴリズムを示す．

3.3. オートマトンの形成

オートマトンは階層性がある．最上位層は，文型パターン辞書と対応する部分で，1パターンが1オートマトンに対応する．名詞句や節などの変数は中位層で定義する．最下位層は前終端の変数や字面で構成されるオートマトンである．なお，中位層以下の深さの差に意味はない．

中位層以下は，共有して参照できる．しかし，文型パターンのレベル分けの意図より，本稿では，粒度の細かい変数がより大きい変数を参照しないこととする．たとえば，名詞句の定義において，埋め込み型となる名詞修飾節は含めない．そのような名詞修飾節は最上位層で解析することを想定する．

3.3.1 アークへの機能の割り当て

アークには，品詞や句の判定のための「条件比較」，活用形や語義の指定のための「制約」，そして，適合解を獲得するための「バインド処理」という3つの機能的属性がある．

1) アークにおける条件比較

入力文が条件を満たすとき，オートマトンの解析状態が遷移する．字面，変数，様相関数，文節境界記号，および，離散記号は，「比較条件」となる．

字面の条件は，原文の表記のほかに標準表記の比較を行う．変数の条件は，単純な品詞コード列等の比較と，サブオートマトンによる比較の２種類がある．それぞれを「S型変数(Simple)」，「A型変数(Automaton)」と呼ぶ．様相関数は，「～られる」や「～ている」など助詞・助動詞および相当語句を判断する．文節境界記号は，入力の参照位置が文節境界か非文節境界かを判断する．離散記号は，修飾語句などが挿入されてもよい場所かどうかを判断する．

2) アークにおける制約

制約条件は，アークの条件比較の結果に対して調べる条件である．比較条件に付随してアークに割り当てる．

語尾関数は変数のバインド値の末尾部分についての制約条件である．制約条件の検査対象は変数ごとに異なる．S型変数の場合，変数照合の直後に制約を検査する．A型変数の場合，A型変数の定義内で指定する部分に対して検査する．たとえば，図2において変数 VP への語尾制約は「^」印付きの変数「 *V* 」に対して行う．

字面関数の指定字面は，関数引数として与えられたパターン記述の照合結果に対する制約条件である．したがって，最上位のオートマトンにて，引数のパターン記述から比較条件のアークを形成し，字面関数の指定する字面はそれらのアークに制約条件として付随する．

3) アークにおけるバインド処理

バインド処理とは，アークの条件比較の結果を記憶することである．比較条件に付随してアークに割り当てる．

マクロ関数の引数に記述されるパターン記述要素列はアークに割り当てる．マクロ関数の指定変数はそれらのアークに付随し，適合したアークを指定変数にバインドする．

記憶記号は，対象となるアークに付随し，適合したアークを記憶する．

3.3.2 記号のオートマトンへの展開

記号の中でも選択記号，任意要素記号，順序入れ替え型の記号は，ノードとアークの結合の仕方で機能を実現する．

特に，順序任意要素指定記号では，順序入れ替え後の全ての組み合わせを展開する．位置変更可能要素指定記号は，対象要素定義「$*n*^{…}」の内容を移動位置「$*n* 」に置き，1回だけその内容が使われるようなオートマトンに展開する．

3.4. 照合アルゴリズム —— AB-ATN

AB-ATN では，文型の適合解の管理のためにエージェントの概念を導入している．エージェントは，ノード上に存在することで，遷移状態を指し示す．そして，遷移過程で得たバインド

値や遷移経路などの適合解を保持する. エージェントの操作には,「 *generate*（新エージェントの生成）」,「 *clear*（全消去）」,「 *breed*（出産・記憶を継承）」,「 *kill*（削除）」,「 *move*（〔一時〕移動）」,「 *sleep*（休眠）」,「 *awake*（覚醒）」がある.

1入力文と1パターンを照合するアルゴリズム AB-ATN1 を図1に示す. 入力文 S と照合するパターン P を入力する. まず, 最初のエージェントをパターンの開始ノードに配置する(2行目). 入力形態素の単位 B で同期をとり全エージェントを動かす(3行目). 覚醒しているエージェントのリストを作り(4行目), 覚醒しているエージェントの存在している間(5行目), 各覚醒中エージェントについてアークを選択させる(6行目). ここで, 複数のアークがあるならば〈アークの数〉-1 個だけ *breed* して, アークと対応付ける. アークを選択したエージェントは Q2 に登録する. 次に, 8, 9行目では, 選択したアーク毎の処理を行う. 比較条件などを満たすエージェントはアーク先のノードに *move* する. jump , pop , サブオートマトンへの移動など, 形態素を使わない場合は, 移動後に再度アークの選択を行うため, Q1 に再登録する. 形態素を使った場合, 使った数だけ *sleep* する. たとえば, 複数の形態素で構成される字面の照合をした場合は, その数のサイクルだけ *sleep* する. また, 条件を満たさなかったエージェントは *kill* される(※1). このサイクルにおいて全てのエージェントが形態素を使うか削除されると, 12行目に処理が移る. clock により時刻が進み, 十分休眠したエージェントは *awake* する. 最後に, 全ての形態素について調べたあとは, 終了動作により途中のノードに存在するエージェントを *kill* する(14行目). そして, 生存しているエージェントの適合解を収集し, 結果として出力する(15行目).

```
 1: function AB-ATN1(S,P)
 2:   Q0 = generate(P)     ※ Pは照合する文型パターン
 3:   foreach B ∈ S        ※ Sは入力文の形態素境界リスト
 4:     Q1 = 覚醒中エージェントのリストの生成(Q0)
 5:     while Q1 ≠ []
 6:       Q2 = アークの選択(Q1,Q0) ※1
 7:       Q1 = []
 8:       foreach C ∈ Q2
 9:         Q1 += アークの処理(C,B,Q0) ※1
10:       end
11:     end
12:     clock(Q0) ※1
13:   end
14:   全エージェントの終了動作(Q0) ※1
15:   return 生存しているエージェントの適合解(Q0)
16: end  ※1.Q0を通じてkill/やsleep等エージェントの状態を管理
```

図1: AB-ATN アルゴリズム

適合解は「アークの処理」の際にエージェントが蓄える. エージェントが *breed* すると, 新しいエージェントが適合解を継承し, 各エージェントが各々の部分適合解を追加していく.

上述の説明は, 1つの入力文と1つの文型パターンを照合するアルゴリズムの説明であった. 1つの入力文と複数の文型パターンを照合するには, パターンを1つずつ照合する方法と, 複数パターンをトライ構造化したオートマトンと照合する方法とが考えられる. 本稿では, 照合するパターンの候補を入力文により動的に絞り込む(4.章)ため, 候補のパターンについて前者の方法で照合する.

3.5. 照合の具体例

次のパターンから作成したオートマトン, および, 使用される名詞句および動詞句のオートマトンを, 図2に示す. なおノード番号は説明の便宜上つけている.

P1: #1〈*NP1*は〉 *VP2*^*katei*ば/*VP3*.*darou*

始点ノードの無いアークが指しているノードが, オートマトンの開始ノードである. 上段はパターン P1 のオートマトンである. 補完要素記号「〈〉」は,「 *NP1* 」と「は」のアークおよび「 jump 」アークで表される. 仮定形の語尾を指定する「^*katei* 」は, かかる変数 *VP2* のアークに制約条件として付与される. 下段左は名詞句のサブオートマトンである. この例では,「の」型名詞句および形容詞修飾つき名詞句が適合する. 下段右は動詞句である.「 *J* 」は格助詞を表す. 開始ノードから動詞の適合により最終ノードに到達できることから, 0個以上の格要素をもつ動詞句が適合する.

ここで, 次の入力に対する照合過程の一部を説明する.

入力文：1太郎2の3弟4は5家6に7帰れ8ば9驚く10だろう11

パターン P1 と照合をするとき, P1 の開始ノードにエージェント$\mathcal{C}_1$を配置(*generate*)する.「アークの選択」において, $\mathcal{C}_1$はノード1でのアークの分岐に従い, $\mathcal{C}_2$を *breed* する. $\mathcal{C}_1$は「 NP1 」アークを, $\mathcal{C}_2$は「 jump 」アークを選択する.「アークの処理」において$\mathcal{C}_1$は NP のサブオートマトンの開始ノードに *move* する. $\mathcal{C}_2$はノード 3 へ *move* する.

再びアークの選択において, $\mathcal{C}_1$はノード 10 で分岐があるため$\mathcal{C}_3$を *breed* し, $\mathcal{C}_1$は「 N 」アークを, $\mathcal{C}_3$は「 AJ 」アークを選ぶ. $\mathcal{C}_2$はノード 3 で「 VP 」アークを選ぶ. アークの処理において, $\mathcal{C}_1$は「 *N*=太郎」に適合してノード 12 に *move* する. 形態素を参照したので, $\mathcal{C}_1$は *sleep* する. $\mathcal{C}_2$はノード 14 に *move* する. $\mathcal{C}_3$は「 *AJ* ≠太郎」のため *kill* する.

アークの選択において, $\mathcal{C}_2$はノード 14 の分岐に従い, $\mathcal{C}_4$を産む. $\mathcal{C}_2$は「 V 」アークを, $\mathcal{C}_4$は「 NP 」アークを選択する. アークの処理で, $\mathcal{C}_2$は不適合となり *kill* する. $\mathcal{C}_4$はノード 10 へ *move* する.

$\mathcal{C}_4$は, $\mathcal{C}_1$と同様ノード 10 で$\mathcal{C}_5$を *breed* し, 自身は「 *N*=太郎」に適合して *sleep* する. $\mathcal{C}_5$は「 AJ 」の不適合で *kill* する.

こうして, 全てのエージェントが入力形態素の1番目について照合をしたので, 1 サイクル目の処理が終了する. この時点で, $\mathcal{C}_1$は *NP1*=「 N(太郎)」, $\mathcal{C}_4$は *VP2*=「 NP(N(太郎))」という部分適合解を持つ.

2 番目の形態素について同様に処理をしたところ, $\mathcal{C}_1$は *NP1*=「 N(太郎),の」, $\mathcal{C}_4$は *VP2*=「 NP(N(太郎),の)」という解を持つ. また, $\mathcal{C}_1$や$\mathcal{C}_4$は,「 pop 」アークへの分岐のためのエージェントを産む. しかし, それらは, pop した後の照合(「は」や「 J 」)で不適合となり *kill* する.

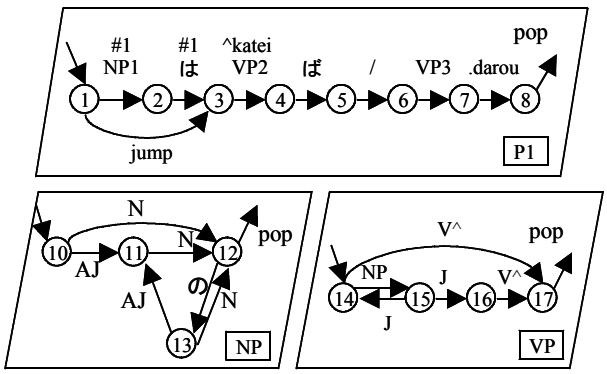

図2: 文型パターン, 名詞句, および, 動詞句のオートマトン

4. 照合候補の絞込みによる高速化

全ての文型パターンと照合するのではなく，見込みの有るパターンに絞り照合することで，高速化が期待できる．

文型の非線形性が字面として表れるというパターンの性質と，必須の字面が入力文に必ず対応しなければならないという適合条件により，パターン中の字面が照合候補を絞り込む手がかりとなる．また，必須字面は様相関数にも含まれている．たとえば「.teiru 」関数は「～てい，～ている，～ていれ，～ていろ」と活用するので，「て」，「い」という字面が必須となる．

本稿では，パターンの字種（字面の文字コード）が入力文に含まれるかどうかで判断する．

5. 実装

5.1. 文型パターンパーサの構成

本パーサは，オートマトン生成系，入力文前処理系，絞込み系，および，照合本体系から成る．オートマトン生成系は，文型パターン記述からオートマトンを生成する．入力文前処理系は形態素解析および入力文中の字種のビットマップを生成する．絞込み系で候補を作り，本体系で AB-ATN が動作する．

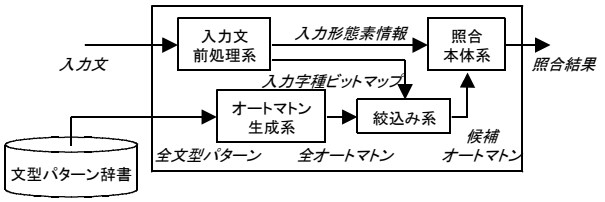

図3： 文型パターンパーサの構成図

5.2. 照合実験

2.2.節で述べた規模のパターン辞書が構築されており，[5], [6]ではこの辞書を用いて，クロスバリデーションテストやオープンテストを実施した．本方式の処理性能を示す．

パターン作成に用いた原文より 1,000 文をランダムで抽出し，各レベルごとの照合性能を測った結果を表3に示す．

表3: 文型パターンレベル毎の照合性能

		単語レベル	句レベル	節レベル
パターン総数		124,158 個	87,685 個	11,280 個
候補数	平均	920.7 個	3,217.1 個	713.1 個
	最大	6,381 個	13,455 個	2,519 個
	最小	32 個	626 個	219 個
(平均絞込み率)		0.7%	3.7%	6.3%
適合数	平均	12.8 個	156.0 個	23.3 個
	最大	258 個	5,852 個	1,912 個
(ヒット率)		1.3%	4.8%	3.3%
1文当たり照合時間	平均	0.42 秒	6.27 秒	0.84 秒
	最大	3.79 秒	167.18 秒	21.55 秒
	最小	0.03 秒	0.26 秒	0.03 秒
(1パターン平均)		0.5 ms	1.9 ms	1.2 ms
初期メモリ使用量		574 MByte	374 MByte	47 MByte

※ CPU=Athlon64, Memory=1GB, OS/Lang.=Linux/C, 処理時間はオートマトン生成系 1 回動作+絞込み系・本体系繰り返し動作. 形態素解析時間は除く.

絞込みにより ATN による処理数を大幅に削減できた．絞込み率（〈候補数〉/〈総数〉）によると，単語レベルの効果が最も高かった．パターン内の字面の種類は，日本語原文の平均文字数が 23.3 文字であるのに対し，単語レベルで 12.6 字種，句レベルで 9.2 字種，そして，節レベルで 7.4 字種であったことが効果の要因である．しかし，ヒット率（〈適合数〉/〈候補数〉）によると，いずれのレベルも候補の絞込みの余地が残されているといえる．関数の必須字面の効果を高める必要がある．

1 文当たりの照合時間は，候補数が多い句レベルで最も時間がかかった．1 パターン当たりの照合時間（〈照合時間〉/〈候補数〉）を求めると，句レベルと節レベルがともに時間がかかった．句レベルと節レベルでは，助詞・助詞相当語（ 3.5.節における変数 J ）に，約 280 語を登録しているため，字面で定義された変数の照合に時間を要したと推察する．

各レベルのパターン辞書をロードした直後のメモリ使用量は，単語レベルが最も多かった．パターン内の文字列の種類数および変数などの構成要素の数が原因である．

6. 今後の課題

今後の文型パターン辞書の開発には,文型の意味カテゴリの付与，意味属性制約の付与が予定されている．

意味カテゴリには，因果，原因理由，時間的推移，などが検討されている[7]．意味属性の付与は，同形のパターンであってもバインドされる動詞や名詞の意味によって，パターンに付与される意味が異なることへの対策である．日本語語彙大系における結合価パターンの意味属性と同様の考え方による．

文型パターンパーサには，変数のバインドに意味制約をかけること，および，意味制約による絞込みの実現が課題となる．

7. おわりに

本稿では，等価的類推思考の原理に基づく機械翻訳方式の実現に向け，文型パターン辞書に対応した文型パターンパーサを試作した．文型記述に対応したアークの設計，および，エージェントにより適合解を管理する ATN アルゴリズムを示した．そして，処理候補の絞込みを行うことで，約 22 万パターンを対象にした照合実験が可能になることを示した．

謝辞

本研究は，科学技術振興機構(JST)の戦略的基礎研究事業(CREST)で行っているものである．

参考文献

[1] 池原ほか:等価的類推思考の原理による機械翻訳方式，信学技報，TL2002-34, pp.7-12, 2002.
[2] 池原ほか:機械翻訳のための日英文型パターン記述言語，信学技報, TL2002-48/NLC2002-90, pp.1-6, 2003.
[3] 池原ほか:非線形な表現構造に着目した重文と複文の日英文型パターン化，情処研報，2004-NL, 2004(3月予定).
[4] 野村:自然言語処理の基礎技術，信学会, 1988.
[5] 池原ほか:日本語重文・複文を対象とした文法レベル文型パターンの被覆率特性，SIG-SLUD, 2004 (3月予定).
[6] 前田ほか:パターンを使用した重文複文の日英翻訳の精度，言処学大会, 2004(予定).
[7] 衛藤ほか:意味類型構築のための接続表現の体系化について，情処研報, 2003-NL-155, pp.31-38, 2003.