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

2006－NL－176（2）
2006／11／22

# 重文・複文文型パターン辞書からの構造照合型パターン検索

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
〒680-8552 鳥取市湖山町南 4-101
E-mail: {tokuhisa, murakami, ikehara}@ike.tottori-u.ac.jp

**あらまし**　日本語の重文・複文の文型パターン 22.8 万件を収録した文型パターン辞書から，解析対象の文と構造の一致する文型パターンを全て検索することを，ATN を用いて実装し，その検索性能について報告する．文型パターンは，「字面」，「変数」，「関数」，「記号」で記述する．記述子に定めた適合条件を柔軟に処理するために ATN を採用した．ATN の高速化のために 3 種類の処理を追加して，トップダウン解析における候補の絞り込み，および，重複する処理の除去を実現した．
**キーワード**　文型パターン，照合，検索，ATN，検索インタフェース

# Pattern Search by Structural Matching from Japanese Compound and Complex Sentence Pattern Dictionary

Masato Tokuhisa　Jin'ichi Murakami　Satoru Ikehara

Faculty of Engineering, Tottori University
Minami 4-101, Koyama, Tottori, 680-8552, Japan
E-mail: {tokuhisa, murakami, ikehara}@ike.tottori-u.ac.jp

**Abstract**　This paper mentions a method that searches all sentence patterns matching with a sentence from a Japanese compound and complex sentence pattern dictionary, which contains 228,774 patterns, and then shows the performance of searching in some experiments. The sentence pattern is described in *orthography*, *variable*, *function* and *marker*. In order to treat these descriptors flexibly, ATN is adapted to the search method. Three functions are added to the method to realize to narrow candidates of patterns for top-down analysis in ATN and to remove the duplicate matching in ATN.
**Key Word**　sentence pattern, pattern matching, pattern search, ATN, interface

## 1. はじめに

機械翻訳は原言語表現の意味を変えずに目的言語表現に変換するという目的により，言語表現の意味を捉える技術に向けて多くの研究がなされてきた．その1つのアプローチとしてパターンを用いる方法が提案されている．

パターンを用いるアイデアは，機械翻訳だけではなく，情報抽出，情緒・感情の解析，言い換えなど多くの研究で取り入れられている．たとえば，情報抽出では，川浪らは，Webドキュメントに複数の抽出パターンを照合し，適合した情報を用いて用語説明を実現している[1]．情緒・感情の解析では，田中らは，情緒の情報を付与した結合価パターンの辞書を構築し，パターンの照合による情緒・感情面での意味理解を目指している[2]．

ここで，パターンの利用は，単なるアイデアではなく，言語表現の意味を捉える技術として本質的な方法であるという考えから，言語表現の意味を扱うための一方式として，意味的等価変換方式が提案されている[3][4]．この方式では，文を単位として表される概念があると考え，文のある一部を抽象化したときに表される概念も対応して一部が抽象化されると考える．翻訳は，原言語の文，その文に表される概念，さらにその概念を表す目的言語の文という3者の関係を追跡することに相当する．機械翻訳を行う上では，3者の間で対応関係のある部分を抽象化した知識ベースを次のように用いる*1：

(1) 抽象化された原言語文の集合（知識ベース）の中から，原言語文と適合するものを検索
(2) 検索結果より，抽象化された原言語文と関係を持つ抽

*1 ピボット方式のピボットと本方式の概念は両言語の間に存在するという点で共通だが，機械処理のために概念を中間表現に書き起こし尽くされることを前提として，翻訳を行うものではないので，本方式はピボット方式ではない．一方，トランスファ方式と比べると，形態素情報のレベルでのトランスファ方式といえる．

象化された目的言語文を選択
(3) 抽象化された部分の翻訳を別途実施
(4) 部分訳の挿入により目的言語文を生成

抽象化された文は，具体的には文型パターンとして表すことができる．たとえば，日本語語彙大系では，日本語の基本的な動詞を中心とした文型パターンと英語文型パターンの対が約 14,800 件収録されている．[3]での日本語の重文・複文を対象とした文型パターン辞書では，約 23 万件が収録されている．

日本語語彙大系のように，用言を見出し語として辞書を構築すると，上記(1)の検索が容易である．しかし，重文・複文の文型パターン辞書は，用言も抽象化したためにパターンに見出し語が無い．そのため，適合するパターンを検索することは容易でなく，原則として，1件1件の文型パターンと原言語文を照合し，適合／不適合を検査しなければならない．

そこで，本稿では，日本語文に適合する文型パターンを文型パターン辞書より検索する方式を開発する．文型パターンの記述子には，品詞を保ちながら抽象化した「変数」(動詞変数 *V* や名詞句の変数 *NP* )，抽象化されなかった「字面」，先行する記述子に作用する「関数」などがある．そこで，機能面の拡張に適した ATN を基礎として，日本語文と文型パターンの照合を行う．ただし，単純な ATN は高速ではない．そこで，本稿では，文型パターンの記述上の特性，および，日本語文と文型パターンの適合条件に着目して高速化を施す．そして，実験によりその動作特性を示す．また，幾つかの応用について述べる．

## 2. 文型パターンの記述と適合条件

文型パターンは，「字面」，「変数」，「関数」，および，「記号」で記述する．これらの記述子は，以下に示すように種類分けされる．文型パターンが文と適合するとは，記述子の指定するとおりに，文の全ての形態素が記述子と対応することをいう．以下には適合の条件もまとめている．

- 字面
  - **生字面**：表記は〈全角文字列〉，適合条件はこの文字列と形態素の文字列の一致である．
  - **終止形字面**：表記は〈’〉〈全角文字列〉〈’〉，適合条件はこの文字列と形態素終止形の文字列の一致である．
- 変数
  - **変数**：表記は〈変数名〉〈変数番号〉，変数名は半角大文字列，適合条件は変数名に定義される品詞と形態素の一致である．適合の結果は変数に代入する．
  - **意味制約付き変数**：表記は〈変数名〉〈変数番号〉〈意味制約〉，適合条件は，変数の適合に加え対応する形態素の意味属性の充足である．
- 関数
  - **語形関数**：表記は〈^記号〉〈関数名〉，関数名は半角小文字列，適合条件は，先行する記述子に対応する形態素の語形と関数に定義される条件との一致である．
  - **様相関数**：表記は〈. 記号〉〈関数名〉，適合条件は，関数に定義される条件と形態素の一致である．
- 記号
  - **離散記号**：表記は〈/記号〉〈離散型〉または〈/記号〉のみ，離散型は半角文字列，適合条件は，離散型に定義される条件と形態素の一致である．離散記号に対応する形態素は，文型パターンによる解析の対象外として扱われる．
  - **文節境界記号**：表記は〈+記号〉または〈！記号〉，適合条件は，前者は形態素間に文節の境界がないこと，後者は形態素間に文節の境界があることである．
  - **まとめ記号**：表記は小括弧“(”と“)”であり，括弧内の記述子をまとめて1つの要素とみなすこととする．
  - **任意記号**：表記は大括弧“[”と“]”であり，括弧内の記述子が適合しても適合しなくてもよいという適合条件にする．
  - **補完記号**：表記は括弧“<”と“>”であり，任意記号と同じ制御機能をもつ．文型パターンを作成する元となった原文に対応する形態素が存在しないことを表す．
  - **順序任意記号**：表記は中括弧“{”と“}”であり，中括弧内はコンマで区切られている．適合条件は，コンマ区切りされた記述子全てが形態素と順序を問わずに一致することである．
  - **移動可能記号**：表記は，〈$記号〉〈移動要素番号〉〈^記号〉〈{記号〉〈記述子〉〈}記号〉および〈$記号〉〈移動要素番号〉である．前者により移動可能な要素をこの記号に割り当てる．後者は割り当てた要素が来ても良いパターン上の位置を表す．適合条件は，前者または後者の位置のいずれか 1 つにおいて，割り当てた要素が適合することである．
  - **選択記号**：表記は〈｜記号〉，適合条件は〈｜記号〉で区切られた記述子のいずれかが適合することである．
  - **記憶記号**：表記は〈＃記号〉〈記憶番号〉であり，後続する記述子に適合した形態素を記憶する．

以下に文型パターンの記述例を示す．

```
$1N1#1(は|が)$1^{ADV2}/cV3^renyou、<N4 は
>{/kN5 を,/kN6 に}'渡す' (.genzai|.kako)。
```

たとえば，「急にピエロが目の前に登場し、子供達に綺麗な花束を渡した。」という文はこの文型パターンに適合する．「急に」という副詞は，左端の$1 の位置で ADV2 と適合する．「ピエロが」は「は」と「が」の選択において「が」が適合し，#1 に「が」だ代入される．「目の前に」は「/c 」と適合するためパターンによる解析の対象外として扱われる．「登場し」は動詞変数 V3 と適合し，連用形の語形関数 ^renyou を満たす．「子供達に」と「綺麗な花束を」は順序任意記号により適合する．「渡した」は「渡し」の終止形が「渡す」であるので終止形字面と適合し，「た」は.kako と適合する．

# 3. 文型パターンの検索

## 3.1. 意味的等価変換方式からの要求

意味的等価変換方式を実現するには，文型パターン辞書から，解析対象とする文に適合する文型パターンを全て検索しなければならない．さらに，変数や記号への代入も可能な組み合わせ全てを検索結果に添えなければならない．たとえば，「/kNP1 」と「太郎の服の色」を照合すると，NP1 への代入は「太郎の服の色」，「服の色」，「色」の3通りが可能である．それは「/k 」に「〈名詞〉〈の〉」が適合するためである．

ただし，変数に代入される内部の解釈の違いを区別する必要はない．たとえば，「太郎の服の色」は「（太郎の（服の色））」と「（（太郎の服）の色）」の2つの係り先の解釈が候補として存在するが，その区別は必要としない．

## 3.2. 検索の方針

本稿は，文型パターンごとにネットワーク化して幅優先 ATN により適合する文型パターンを検索するという方針をとる．以下に理由を述べる．

### 3.2.1 ATN を用いる理由

検索では，解析対象の文をキーとして，検索で求めるものはその文に適合する文型パターンおよび変数等の代入値である．文中の文字列は変数に適合することがあるので，文中の文字列をインデックスとする検索はできない．ゆえに，本稿では，文型パターンを1つずつ辞書から取り出して文と照合することで，適合パターンを見つける．

文型パターンと文との照合は ATN を用いる．変数と文の照合は，構文解析と同じく形態素列を検査するからである．一般に，ATN よりもチャート法が高速である[5]が，本稿では次の理由から ATN を用いる：

（a1）記述子の複雑な適合条件に柔軟に対処可能

（a2）パターン辞書を句構造規則集とみなしたとき，チャート法では，active arc の管理コストが高い

チャート法が ATN に勝る点は，重複する条件検査がチャート法では抑えられている点である．ATN で文型パターンの変数，様相関数，離散記号の条件検査をする際，サブネットワークを用いて照合するため，この部分の重複処理を除去する必要がある．

### 3.2.2　文型パターン毎のネットワーク化の理由

ATN のネットワークを作成する単位として，原則として，1つの文型パターンに1つのネットワークとする[*1]．複数のネットワークをトライ構造でまとめることは可能であるが，次の理由からネットワークをまとめることはしない：

（b1）文型パターンはバリエーションが広いためトライ構造で共通化できる部分は多くない．

（b2）文型パターンには字面の記述子がある．任意記号や選択記号の係らない字面は，文が適合するための必須の条件であるので，ATN を用いる以前に文の不適合を判断することができる．

### 3.2.3 幅優先の理由

ATN において状態遷移を行う際，曖昧性が生じる．この対策として，並列に状態遷移を行う方法（幅優先），および，状態の分岐を記憶しておき，一方を行う方法（深さ優先）がある．次の理由から幅優先とする：

（c1）文型パターンのネットワークは一直線になりやすく，選択記号による分岐のあと，再び同じノードにアークが結びつく．

（c2）文型パターンの曖昧性は，選択記号による分岐の他に，変数や離散記号に適合する文の部分の違いによるものである．このために生じた複数の状態遷移は，後に同一の状態遷移を起こしやすい．

これらの性質は，後述する join 動作が効率的に動作することを示すものである．

## 3.3. ATN を用いた検索

ATN によるパージングの基本的な説明は他の文献に譲る．文型パターン辞書や記述子による特殊な部分について説明する．

### 3.3.1 AB-ATN

ATN を幅優先で実行するために複数の遷移状態を管理することを，本稿では「エージェント」という喩えをとることで簡単に説明する．ゆえに，本稿ではこの ATN を AB-ATN（Agent based Breadth first-ATN）と呼ぶ．

エージェントは，状態遷移の過程で得た変数などの代入値を保持し，push や pop の動作に対処するためのスタックを持つ．また，語形関数や意味制約のための記憶も持つ．複数のエージェントの動作の同期は，文を構成する形態素の参照位置とする．

エージェントの動作は次の 7 種類である．

- **move 動作：** エージェントはネットワーク上に存在する．アークには記述子の適合条件があり，基本的に参照する形態素と比較する．その条件が満たされるときエージェントはアークをたどり次ノードに移動する．push と pop の動作は広い意味でここに位置づける．
- **kill 動作：** 条件が満たされないときエージェントは消滅する．
- **memorize 動作：** 変数のアークや記憶記号付き記述子のアークを移動した後，対応する形態素を記憶する．
- **sleep 動作：** move 動作の後，記述子によっては2つ以上の形態素と適合することがある．形態素の参照位置

*1 移動可能記号については記号を展開した文型パターンに変換するためこの限りではない．

で幅優先探索の同期をとるため，先読みをしたエージェントは差分の時間だけ休眠する．

- **awake 動作：** 形態素の参照位置により同期がとれるときエージェントは目覚める．
- **breed 動作：** 1つのノードから複数のアークが出ている際，エージェントは子供を産み，各アークに配置する．親エージェントの記憶を子供エージェントが引き継ぐ．
- **join 動作：** ネットワーク上の同一ノードでエージェントが重なるとき，記憶を適切に引き継ぎ1つのエージェントになる．詳細は高速化の節で説明する．

### 3.3.2 記述子の適合条件の検査

記述子の条件を検査する方法について，選択記号，任意化記号・補完記号，変数・様相関数は，通常の ATN と同様にアークの分岐，jump アーク，サブネットワークを用いる．ここでは，字面と語形関数・意味制約を説明する．

形態素解析において曖昧性がある場合，複数の形態素（品詞コード，終止形，活用形）を出力する．たとえば，「とります。」は「取る」，「撮る」などの終止形の曖昧さと「取り」，「撮り」などの活用形の曖昧さがある．また，「最中」は「時詞」と「名詞（モナカ）」の品詞の曖昧さがある．

「終止形字面」の照合では，「’取る’」という記述子に対して「とります」の「とり」は上記の形態素情報に基づき適合する．また「取ります」という記述子に対しても同様である．

語形関数や意味制約は，変数に代入された形態素のある部分と照合する．たとえば，動詞句の変数には，「〈格要素〉〈用言性名詞〉を〈する〉」の品詞が適合するが，語形関数は〈する〉の部分に制約をかける．また，意味制約は〈用言性名詞〉の部分に制約をかける．そのために，エージェントは，語形関数と意味制約の制約先の情報を記憶しておく．制約先を明示するために変数の定義においてマーカをつけておく．たとえば動詞句の定義を簡単に説明する：

```
1  VP → CSC ND* を 'する'^
2  VP → CSC V^*
```

マーカ「^」は語形関数の制約先を表す．マーカ「*」は意味の制約先を表す（クリーネ・スターではない）．VP は動詞句，CSC は格要素列，ND は用言性名詞，V は動詞である．エージェントが VP のサブネットワークにおいて，マーカと記述子のあるアークを通過するとき，記述子に対応した情報を記憶する．

## 3.4. 高速化

AB-ATN には3つの点で高速化の余地がある．

**(1) 絞り込み前処理：** 文型パターンに必須の字面がわかっている．文中にその文字が存在しないならば，ATN を使うまでもなく不適合がわかる．語順を問わず文字のビットマップ化により数バイトの検査で判定できる．

**(2) join 動作：** 目覚めているエージェントは，参照する形態素位置は同一である．文型パターンのネットワーク上で，複数のエージェントが同一のノード上に存在し，かつ，記憶している制約先の形態素が同一であるならば，その後の状態遷移は同一である．ゆえに，それらのエージェントは合体して1つになる．

**(3) サブネットワーク照合の履歴：** 変数，様相関数，離散記号は，サブネットワークを用いて照合する．そこで，その照合結果の履歴を残すことにする．履歴の検索キーは，サブネットワークの種類（たとえば変数名）と参照する形態素位置である．履歴は，文が同一ならば文型パターンに関係なく利用できる．

# 4. インプリメンテーション

構造照合型文型パターン検索システムの構成図を示す．絞り込み処理および AB-ATN を Linux 上の C 言語を用いて実装した．文型パターン辞書は，ネットワーク化した後，本システムの主記憶に一括ロードする．変数定義のためのサブネットワークも同様である．

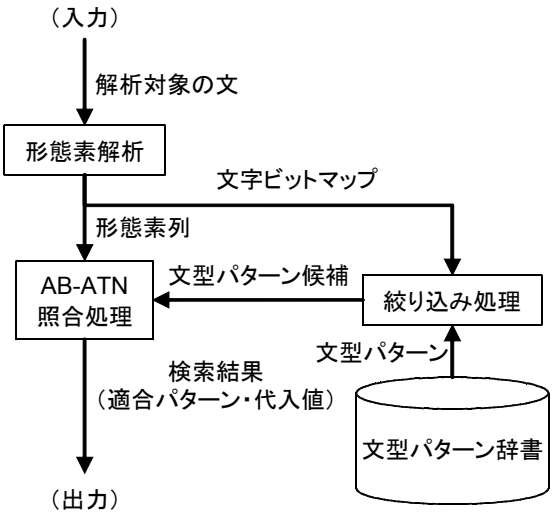


図1 構造照合型文型パターン検索システムの構成

# 5. 実験

提案した文型パターン検索システムの動作特性を計測することを目的とする．

## 5.1. 実験環境

使用する計算機は，CPU が AMD Athlon64 2.4GHz，メモリが 2GB，OS が SUSE-Linux（64bit）である．

検索でキーとして与える文は，日本語が重文・複文である15万文対の日英対訳コーパスから取り出した日本文である．文型パターン辞書はこのコーパスから作られたが，検索の際，与える文から作られた文型パターンは照合しないこととする．

文型パターン辞書は，単語レベル，句レベル，節レベルの 3 レベルから構成される．変数の最大の大きさでレベル分けされている．本実験では，単語レベルのパターン集（122,718 パターン）を用いる．なお，変数・離散記号は 12・17 種類であり，255 個の規則がある．

## 5.2. 基本的な動作特性

履歴の利用による ATN は $O(n^3)$であると言われている[5]. また, 幅優先探索では状態数が問題になる. そこで, この2点について本システムの動作特性を調べる.

### 5.2.1 実験1:文の長さと検索時間の関係

文型パターン辞書(単語レベル)から文に適合するパターンを全て検索する時間を計測する. コーパスから文の長さ(単語数)ごとに 200 文ずつ抽出した 5,527 文を用いる. 長さが 29 以上の文は 200 文に満たなかったので, 得られた文全てを用いた. 図2に実験結果を示す. このグラフには, パターンが検索できた場合(ヒット)とできなかった場合(ミス)についての平均検索時間, および, それらの総合の平均検索時間を載せている. 次のことがわかる:

- ヒットの場合とミスの場合では, ミスの場合が僅かに時間が短いが, 大差がない.
- 200 文まで抽出できなかった長さの文(長さ 29 以上)では, 2つの場合とも検索時間のばらつきが目立つ.
- 長さ 28 以下について総合の場合のプロットを近似式で表すと, $y = 0.0004x^2 + 0.004x - 0.007$ となった. 文の長さ $m$ に対して, 実効的には, $O(m) \sim O(m^2)$ である.

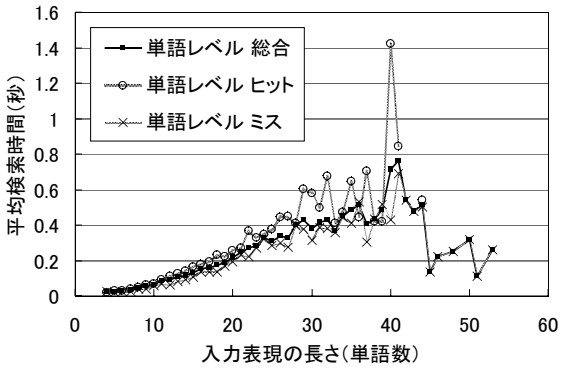


図2 文の長さと検索時間の関係

### 5.2.2 実験2:辞書の規模と検索時間の関係

単語レベルパターン辞書を 10 分割して, 徐々に用いるパターンを増やしながら, 辞書の規模ごとの平均検索時間を調べる. コーパスからランダムに取り出した 6,181 文を用いる. 結果を図3に示す. また, プロットした点の近似式を求めた. 次ことが読み取れる:

- パターン辞書の規模 $p$ (パターン数)に対して平均検索時間は, $O(p)$である.
- パターン辞書の規模が大きくなるにつれて, 1パターン当たりの検索時間が少ない. たとえば, 規模が約 50,000 の際の時間と約 85,000 の際の時間のそれぞれを近似式と比べる. これは, 辞書の規模が多くても不適合パターンの割合が多くなるためと考えられる.

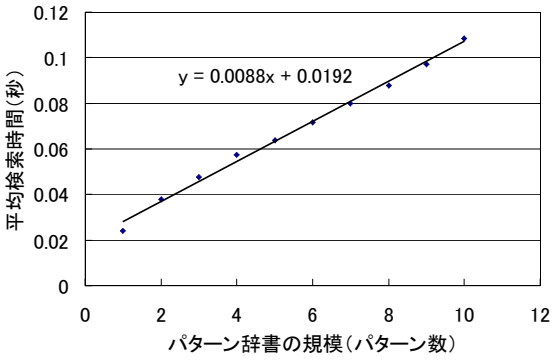


図3 辞書の規模と検索時間の関係

### 5.2.3 実験3:文の長さとメモリ使用量の関係

メモリ使用量は, 遷移状態(エージェント数), 履歴(キー, 値), および, 変数等への代入値が主たる要因となって決まる. 実験1と同じ文集合を用いて検索実験を行い, 文の長さに対するメモリ使用量を計測する.

表1に, 各要因について, 実験で使用された個数の最大値, 1個当りに使用するメモリ( byte 単位), 実験で使用された最大値・平均値( Mbyte 単位)を示す. 図4は, これらの合計値を文の長さごとに集計した平均値を示す. 表1と図4より次のことが言える:

- 遷移状態は1個当りのメモリ使用量は大きいが, 全体でみるとメモリ使用量に占める割合は大きくはない.
- 代入値のために使用したメモリ使用量が最も大きい. 照合の途中で失敗した場合でも, 使用した代入値は履歴に残すために破棄しなかったためである.
- 使用メモリ量の平均値(図4)と最大値(表1)を比較すると, 平均使用メモリ量の主要因は「代入」である.

以上より, 現在の計算機の能力から言えば, 本システムは現実的な処理能力であるといえる.

表1 各要因のメモリ使用量

| 要因 | 個数 | | 使用量 | |
|---|---|---|---|---|
| | 最大 | 1個当り | 最大 | 平均 |
| 遷移状態 | 576 | 648 | 0.36 | 0.008 |
| 履歴:キー | 1,337 | 8 | 0.01 | 0.003 |
| 履歴:値 | 3,492 | 24 | 0.08 | 0.020 |
| 代入 | 3,318,437 | 12 | 37.98 | 2.888 |

※ 使用量の単位は1個当りが byte , 他は MB

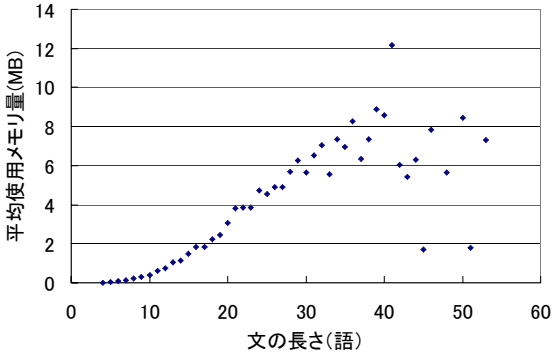


図4 文の長さと使用メモリ量の関係

## 5.3. 高速化の効果

第 3.4.節で示した3つの高速化の効果を調べる. 実験1と同じ文集合を用いて実験する.

### 5.3.1 実験4:絞り込みの効果

絞り込みの条件は, 文型パターンの中で適合することが必須の字面である. そこで, まず, 単語レベル文型パターン辞書における必須字面を集計した. 必須字面の種類数は, 93,792 種類であり, 使用頻度が 1 であるものは 88,303 種類, 2 であるものは 3,335 種類であった. 逆に使用頻度の高いものは表2のとおりであるが, 約 12 万の単語レベルパターンにおいて,「。はを」の字面を必須字面とする文型パターンは 1%に満たない. こうした, 必須字面の特殊性により絞り込みの効果が期待できる.

表2 単語レベル文型パターン辞書において使用頻度の高い必須字面(上位10件)

| 順位 | 必須字面 | 頻度 |
|---|---|---|
| 1 | 。はを | 1,127 |
| 2 | 。は | 909 |
| 3 | 。を | 708 |
| 4 | 。には | 559 |
| 5 | 。とは | 467 |
| 6 | 。にあを | 459 |
| 7 | 。のはを | 457 |
| 8 | 。が | 450 |
| 9 | 。がは | 365 |
| 10 | 。にを | 321 |

次に, 絞り込み機能を無効にして文型パターンを検索する. 文の長さごとの平均検索時間を求める. 実験1と比較した結果を図5に示す. この結果より次のことが言える:

- 検索時間に大きな開きが見られる.
- 文の長さが 16 語を超えたあたりから, 絞り込み無しの場合の平均検索時間の増加が鈍くなっている.

ここで, 後者の理由を調査すると, 処理の途中でアボートが生じていることが分かった. 長さ 16 語以上の入力において, アボートが見られ始め, 長さ 30 語の文集合においては 37%(99/156)の文がアボートした. アボートの理由はメモリ不足である. 不適合でも本システムでは代入のためのメモリを使用するためである.

また, アボートしなかった文の実行結果を調べると, 平均検索時間は, 絞り込み無しの場合, 1.449 (秒/文), 絞り込み有りの場合, 0.225 (秒/文)であった. 絞り込みにより検索時間は約 6 分の 1 に短縮できた.

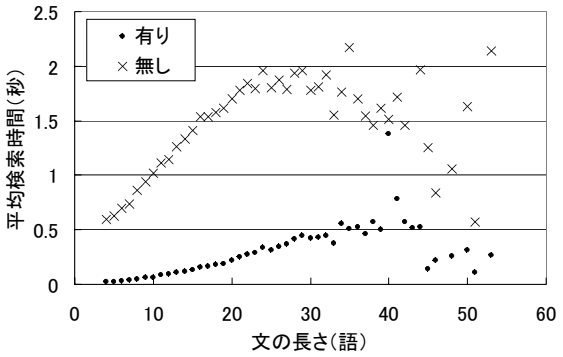


図5 絞り込みの有無による平均検索時間の比較

### 5.3.2 実験5: join 動作の効果

join 動作をさせない場合もアボートが発生した. アボートせずに動作した 99 文について動作状況を調べる. その結果(表3)から次のことが言える:

- エージェント数(遷移状態の並列数)は, 最大値で 20 分の 1 (8,356/163,901)に, 平均値で 5 分の 1 (1,833.2/8,688.6)に, それぞれ抑えることができた.
- 検索時間は, 9 分の 1 に短縮できた.

表3 join 動作の有無による処理性能の比較

| 検査項目 | join 動作あり | join 動作なし |
|---|---|---|
| join 回数 | 788,127 回 | − |
| 1 ネットワーク当り*1 | 1.23 回 | − |
| エージェント数*2 | | |
| 最大値 | 8,356 個 | 163,901 個 |
| 最大値の合計 | 181,489 個 | 860,171 個 |
| 1 文当り平均 | 1,833.2 個 | 8,688.6 個 |
| 検索時間 | 10.9 秒 | 93.3 秒 |

*1 照合処理をしたネットワークの数で join 回数を割る
*2 瞬間的に並列的に存在した数

### 5.3.3 実験6:履歴機能の効果

検索処理がアボートした回数を求めた(表4). 長さが 16 以上になるとアボートの発生が始まり, 長さが 30 になると 57 回( 156 回中)となった. アボートの理由はメモリ不足である. また, アボートせずに検索できた場合の検索時間を比較したところ, 38.3 倍の高速化ができたと言える.

表4 履歴の有無による処理性能の比較

| 処理項目 | 履歴機能あり | 履歴機能なし |
|---|---|---|
| 文数 | 5,527 文 | |
| アボート率 | 0.036% | 78.1% |
| アボート数 | 2 文 | 4,319 文 |
| 文数 | 1,208 文 | |
| 検索時間 | 49.74 秒 | 1904.93 秒 |

# 6. 応用

## 6.1. 文型パターン辞書の照合実験

文型パターン辞書は，日英対訳コーパスを変数化や関数化を半自動的に施すことによって構築され，言語アナリストによるチューニングが進められている．

文型パターン辞書の開発過程において，次の2つの照合実験が行われる：

**(1) 自己照合実験：** 文型パターンが正しく汎化されていることを確認するために，文型パターンとその作成元となった日本語文とを照合する（1 文対 1 パターンの照合）．

**(2) クロス照合実験：** 文型パターン辞書のカバー率を確認するために，コーパスの日本語の各文について，その文から作られたパターンを除き適合する文型パターンを辞書から検索する（12 万文対 12 万パターンの照合）．

自己照合実験では現状で表5の結果を得た．総数は作成したパターンの数，適合数はパターン作成の原文と適合したパターンの数である．マッチ率はその比である．100%になっていない理由は，変数の適合条件の定義に誤りがある場合，および，不適切な変数化などである．この実験結果より，その後の文型パターンの修正と変数定義の見直しが行われる．

表5 自己照合実験の結果

| 項目 | 単語レベル | 句レベル | 節レベル |
|---|---|---|---|
| 総数 | 122,719 | 80,417 | 25,638 |
| 適合数 | 121,510 | 76,603 | 23,553 |
| マッチ率 | 99.0% | 95.3% | 91.9% |

（単位はパターン数）

クロス照合実験では現状で表6の結果を得た．適合文数 $I_m$ は，文型パターン辞書による解釈が可能な文の数である．$R1$ はそれを割合で表したものである．適合パターン数 $M$ は文に適合したパターン数であるので，$N_{match}$ より辞書による解釈の多義数がわかる．

節レベルでは非常に多くの多義が存在するが，幾つかの日英パターン対は同一である可能性があるので，節レベルの多義数は縮小する余地が残されている．

$R1$ や $N$ などの値は，変数・離散記号の定義により変化する．現状では，節変数 $CL$ の定義に不完全なところがあるため，この結果は参考値である．具体的には，名詞述語文（ダ文）の解析規則である．名詞述語文の適宜において，判定詞（です，だ，である）が伴うことを必須の条件としていないため，名詞句に $CL$ が適合してしまう．ゆえに，定義の改良により $R1$ は低下するものと予想される．

所要時間の内訳に関して，本実験では，12 万文を検査に用いているが，1 文の形態素情報を 1 ファイルに格納し，その文に対する検索結果も 1 文に対して 1 ファイルに出力しているため，ハードディスクのアクセスに関する処理時間が無視できない．参考として 12 万文について，単語レベルの検索結果を集計するプログラムに約 3 時間かかることから，本実験においても同程度の影響があると考えられる．

表6 クロス照合実験の結果

| 項目 | 単語レベル | 句レベル | 節レベル |
|---|---|---|---|
| 検査文数 $I_{in}$ | | 123,618 | |
| 適合文数 $I_m$ | 80,030 | 102,751 | 118,721 |
| 適合パターン数 $M$ | 2,788,632 | 27,111,868 | 145,043,999 |
| $R1 = I_{in} / I_m$ | 64.7% | 83.1% | 96.0% |
| $N_{all} = M / I_{in}$ | 22.6 | 219.3 | 1,173.3 |
| $N_{match} = M / I_m$ | 34.8 | 263.9 | 1,221.7 |
| 所要時間 | 10h 47m | 10h 34m | 17h 57m |

## 6.2. 日英対訳文の検索

図1で示したシステムに，英語パターンおよび日英パターンの作成に用いた原文を表示するインタフェースを加えることで，日本文を入力し，その文と構造的に類似する日英対訳文を検索結果に出力するという英文作成の支援システムが構築できる．

図6にその実行例を示す．4 つのウインドウが表示されている．上のウインドウでは，文型パターン辞書のレベルおよび検索キーとなる日本文を入力する．ここでは単語レベルが選択され，その規模が表示されている．

中のウインドウは検索結果である．文型パターンは 36 種類が見つかり，代入の違いから 45 通りの結果が得られている．検索結果はパターンが文をカバーする割合が大きく，字面が多く適合する照合結果を優先に列挙する．ここでは，「僕は車を運転するのが上手です。」に対して，「私は絵をかくのがへたです。/I am bad at drawing pictures 」というコーパス中の日英文対が得られている．パターンの変数と文の対応関係は，マウスを変数に載せることで色反転により強調表示される．

右下のウインドウは，英語パターンを用いて英文を生成する．英語パターンはユーザが調整可能である．ここでは，中のウインドウの第一検索結果を利用し，「 *N*1 @be *AJ*3 at *V*3^*ing N*2 . 」という英語パターンで「 I am good at driving wheel . 」という英文を生成した．「 be 」は「@be 」と調整した．英文の訳語選択は，英語言語モデルとして 3-gram を使用してスコアを与えた．訳語「 wheel 」のようにベストではない結果が現状では選択される．

左下のウインドウは，実行状況のモニタである．辞書からの検索時間（形態素解析時間は除く）が 0.38 秒，そして，翻訳結果のスコア（確率の対数）が「−61.1370765240094 」であったことがわかる．

# 7. おわりに

等価的意味的変換方式の実現に向けて，構造照合型文型パターン検索を実装し，その性能評価を行った．十万件規模の文型パターン辞書から，解析対象の文と適合す

る文型パターンを検索することを，ATN を用いて実現したところ，「絞り込み」，「join 動作」，「履歴」の 3 つの高速化により期待通りの動作が確認できた．また，文型パターン辞書の開発に，本検索システムが活用できたことを示した．

本検索システムは，十万件規模の文型パターンを高速に照合できること，さらに，変数へは統語的・意味的な制約をかけることができることより，[1]や[2]などの関連研究におけるパターンマッチングに利用可能である．今後，関連研究に応用し，より現実的なテキストに対する問題点を見い出し，意味処理技術としての改良を試みたい．

## 謝辞

本研究は，独立行政法人科学技術振興機構(JST)・戦略的創造研究推進事業(CREST)の研究領域「高度メディア社会の生活情報技術」の研究課題「セマンティックタイポロジーによる言語の等価変換と生成技術」の支援によるものである．

## 参考文献

[1] 川浪理恵子，大熊智子，増市博，杉原大悟，石崎俊：ウェブからの情報抽出システムの構築－定義型質問に対する情報検索に基づく回答の作成－，言語処理学会第 12 回年次大会発表論文集，pp.797-780，2006．

[2] 田中努，徳久雅人，村上仁一，池原悟：情緒生起情報付き結合価パターン辞書の開発，言語処理学会第 12 回年次大会発表論文集，pp.1151-1154，2006．

[3] 池原悟，阿部さつき，徳久雅人，村上仁一：非線型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，自然言語処理， Vol.11，No.3，pp.69-95，2004．

[4] 池原悟，阿部さつき，竹内奈央，徳久雅人，村上仁一：意味的等価変換方式のための重文複文パターンの統語的意味的分類体系について，情報処理学会研究報告，自然言語処理，2006．

[5] James Allen: Natural Language Understanding, The Benjamin/Cummings, 1994.


図6 日英対訳文検索システムの動作例