知能と複雑系 114－4
（1998．10．2）

構文解析と意味解析の動的な相互作用の実現

北野 智丈　　赤間 清　　宮本 衛市

北海道大学 大学院 システム情報工学専攻

札幌市北区北 13 条西 8 丁目．Tel. 011-706-6815

{kitano,akama,miya}@complex.eng.hokudai.ac.jp

従来の自然言語処理では、構文解析と意味解析が逐次的に行なわれるのが普通である。しかし、その方法では処理の柔軟性に欠け、処理効率に限界がある。

本研究では「等価変換に基づくシステム設計方法」に従って、構文解析、意味解析の統合処理を行う自然言語理解システムを構築した。このシステムでは各解析を制約に基づいて細かく分け、処理順序はコンパイル時に決定されるのではなく、実行時に動的に決めている。実例として文章の解析の過程を示し、柔軟な処理が実現できることを確認する。

自然言語処理　　等価変換　　統合処理　　相互作用

Realization of dynamic interaction of Syntactic and Semantic analyses

Tomotake Kitano　　Kiyoshi Akama　　Eiichi Miyamoto

Division of System and Information Engineering, Hokkaido University

North 13 West 8 Kita-ku Sapporo. Tel. 011-706-6815

{kitano,akama,miya}@complex.eng.hokudai.ac.jp

In conventional natural language processing, syntactic and semantic analyses are executed sequentially. The sequential method is, however, inefficient due to the inflexibility of computation.

In this research, we adopt a system design method based on equivalent transformation to construct a natural language understanding system, which execute integrated processing between syntactic and semantic analyses. In this system, many constraints are used for formalization and processing order is decided, not at the compile time, but dynamically at the execution time depending on the current constraints. We demonstrate flexible processing by explanation of a sample process of computation.

natural language processing, equivalent transformation, integration, interaction

1 はじめに

従来の自然言語処理システムの多くは、形態素解析、構文解析を行い、その結果をもとにして意味解析を行なっている。しかし、この方法では処理の柔軟性に欠け、処理効率には限界がある。なぜなら、多くの文章は構文的に曖昧性を持つため、可能な構文解析結果は多数ありうるが、そのうちの多くは意味的制約を満たさないので最終的には捨てることになり、結果として無駄な意味解析を行なうことになるからである。

特に、未知単語を含んだ文や文法的に不完全な文を扱うとき、取り得る構文構造は膨大な数となり、計算量が爆発的に増加する。よって、従来の統語知識による構文解析に重きをおいた構文主導の解析方式では未知単語を含んだ文などに十分な対応ができないと考えられる。

構文解析と意味解析を相互に関連づけ、お互いに補い合って処理を行えば、より高度な解析を行うことが可能となる。そのためには逐次的な処理ではなく、動的に解析順序を決定できるようなアルゴリズムを書く必要があると考えられる。しかし、従来のシステム設計法では、このようなアルゴリズムを記述するのは困難である。

本研究ではこの困難を克服するために、「等価変換に基づくシステム設計方法」を採用する。この方法で作られたシステムでは各解析をさまざまな制約処理の集合としてとらえ、各制約処理を等価変換ルールで表現し、ルールの適用による制約処理を行うことで解析を進める。適用ルールの選択はシステムにより、計算の状況に従って動的に決定される。それによって、解析順序を固定することなく、確実な情報から処理することができ、構文解析と意味解析が相互に影響しあい、バランス良く処理することができる。人間があらかじめ処理の順序を決定してアルゴリズムを書く必要が無いので、従来のシステム作成方法による困難が回避できる。

本論文では、「等価変換に基づくシステム設計方法」により作られた自然言語処理システムにおいて、構文解析と意味解析の統合処理がもたらす相互作用を説明し、柔軟な処理による解析のコストの減少をデータとして示す。

以下、第2章では、等価変換の理論とその計算の枠組みについて述べ、第3章では構文解析と意味解析の相互作用の仕組みを解説する。第4章では例文を用いて実際にどのような流れで解析が行なわれているか説明し、第5章ではまとめを述べる。

2 等価変換による計算

等価変換による計算では、問題を「宣言的プログラム」[1] を用いて記述し、それを「等価変換ルール」を用いて変換する事により解を得る。ここではその基本的な枠組について述べる。

2.1 宣言的プログラム

宣言的プログラムとは、論理プログラムを一般化したもので、(一般化されたアトムを持つ) 確定節

$H \leftarrow B_1, \cdots, B_n$

の集合である。H をヘッド、$B_1, \cdots, B_n$ をボディと呼ぶ。アトムは制約の表現として用いられる。

2.2 等価変換ルールと計算の制御

宣言的プログラムには宣言的意味が与えられる。等価変換とは、プログラムの変換の前後で宣言的意味が保存されるような変換のことである。ルールは宣言的プログラムのボディアトムの1つに対して作用し、変換を行なう。ルールのシンタックスは次のようになる。

```
(Rule  <ルール名>
    (Head  <ヘッド>)
    (Cond  <ルール適用条件列1>
                :          )
    (Body  (exec  <実行列1>
                  :     )
           <ボディ1>
              :              ))
```

Head はルール適用対象の形式、Cond 部はルール適用条件、exec 部はルール適用時にアトムに対して行なう操作、Body 部はルールが適用されたアトムと置換されるアトム群である。Body 部が空ならば、ルール適用後、そのアトムは消滅する。詳しいルールの適用は次節で述べる。

また、各ルールには優先度を指定することができる。システムは適用するルールを優先度の高いモノから選択するので、ユーザは優先度の付与により計算の制御をすることが可能になる。

他に等価変換の利点として、

- 豊富なデータ構造を提供することで、表現豊かなルールを記述できる。
- 計算の順序が固定されないので、処理を柔軟に制御できる。
- 解の正当性が保証できる。

などがあげられる。

2.3 計算の流れ

システムは、計算対象となるアトムの1つに対し、等価変換ルールの中の1つを適用する。適用されるのは、

- Head とアトムのマッチングが可能であり、
- Cond の適用可能性の判定が成功する
- 優先度の高い

ルールである。ルールが適用されると、まず Body 部の exec 部が実行される。これにより変数への代入などが行なわれる。そして最終的に、ルールが適用されたアトムはボディに記述してあるアトムに置換される。

ここまでが1回の等価変換ルール適用の流れである。計算全体の流れとしては、適用できるルールがなくなるか、アトムがなくなるまでこれを繰り返す。変換アトム及び適用ルールはシステムにより動的に決定される。アトムはそれぞれが独立しているが、共通変数を持っているので、あるアトムにルールが適用され変数の代入などが起きると、それが他のアトムにも伝わりルールが適用されやすくなり、連鎖的なルールの適用が起きる。このような相互作用により、柔軟な計算が実現できる。

3 構文解析と意味解析の相互作用

本システムでは柔軟な処理を実現するために、各解析を細かな制約に分け表現する。制約には文法制約などの構文による制約や、単語の意味や状況などの意味による制約などがある。構文制約も意味制約も同じ制約集合に入れ、全く同じように扱う。制約の処理は、等価変換ルールにより行なう。等価変換ルールは、適用条件、処理手続き、変換後のかたちから成っている。アトムの変換後のかたちは、消滅したり、分裂したりと様々だが、制約集合全体の意味は、正当性が保証された等価変換ルールを用いる事により、保存される。

このシステムでは、たくさんのルールが制約の集合を監視していて、ある制約があるルールの適用条件を満たすとそのルールが適用され、ルールの記述にある処理を実行し、変換を行なう。この処理により得られた情報は、その情報を必要とするすべての制約に伝播される。伝播した情報により別の制約の適用条件が満たされ、ルールの適用による変換が行なわれ、その情報がまた他の制約に伝播する。これを繰り返し行ない、連鎖的に制約の処理を行なう (図1)。本システムでは情報の円滑な授受のために、情報付き変数を用いる。これについては次章で説明する。

このように、構文解析と意味解析を統合し、情報の授受を行なうことにより、相互作用が期待できる。例えば、構文解析で複数の解釈がある場合、従来の逐次的な処理を行なうシステムでは、依存先候補の場合分けを行ない、それぞれについて意味解析を行なう。しかしその中の多くは意味的な制約を満たさないために捨てられることになる。本システムでは依存先が明白な部分から先に意味解析を行い、それにより得た情報を他の意味制約や文法制約などに伝搬させる。それによって意味的には取り得ない係り関係を除くことができ、文脈の曖昧性が減少し、早い段階で依存先候補を絞り込むことができるので場合分けの回数が減り、無駄な処理を省く事ができる。4.5でこれに関する実験結果を示す。

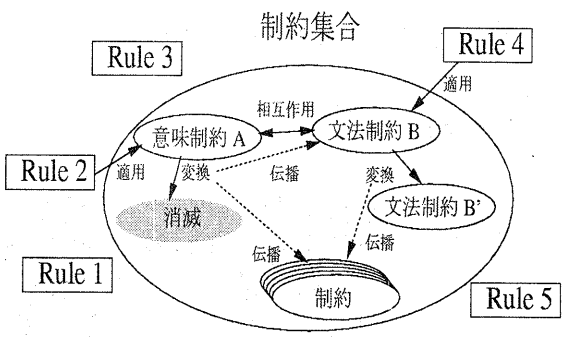

図 1: 相互作用

また、このアーキテクチャは、未知単語を含んだ文の意味理解などの複雑な意味処理を行なう時にも有効である [2]。

4 例題の解析

本章では「先手の金を取れる飛車のとなりの歩を動かせ」を例文として用い、実際にシステムを用いて解析を行った結果、どのように例文が処理されたかを説明する。盤面の情報として図4を与える。

この例文のおおまかな解析の流れを図2に示す。この処理の流れは、システムが解析状況に応じて適用ルールを選択した結果として得られたもので、あらかじめ与えた順序固定のアルゴリズムによるものではない。以下、この過程を大きく4つに分け、処理の流れを順に説明する。

4.1 Step1

システムに文が入力された時、適用可能なルールは形態素解析ルールのみである。形態素解析ルールには以下のようなものがある。

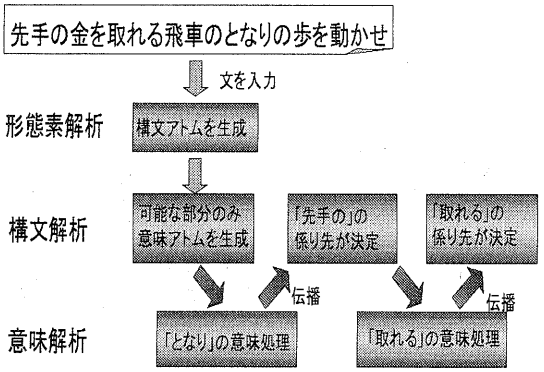

図 2: 解析の流れ

- 辞書引ルール

  辞書の単語とのマッチングにより、文を単語ごとに切り分け、単語の情報を与えていく。

- 文節作成ルール

  単語に分けられた文から自立語と付属語のセットをつくり、文節にまとめる。

各文節は辞書引の際に与えられた情報をそれぞれ付加され、例えば、先の例文では、全文は次のようなリストで表現される。

```
<全文> =
(*1~((単語  先手) (付属語 の) (依存先 *11)
     (意味 *P~(指手 先手)))
 *2~((単語  金) (付属語 を) (依存先 *22)
     (意味  *B~(駒 (種類 金))))
 *3~((単語  取) (付属語 れる) (依存先 *33)
     (意味  (get *A1 *A2)))
 ...
```

本論文ではアトムはS式で記述する。アトムを表すリストの先頭のシンボルが述語名、続く引数が引数である。先頭が「*」がついている文字列は変数である。変数の後ろに「~」がついているものは、情報付き変数である。後ろのS式を “情報” と呼び、変数から情報を取り出したり、取り替えたり、削除したりすることができる。変数が持つ情報の意味はユーザが定義する。

辞書引によって与えられる単語の情報の中で、自立語にはその単語の一般的な意味が意味情報の値として付加される。例えば文節「取れる」には、何か取る物があり、それに対して取られる物があるという意味が存在する。それを get を述語名とし第一引数で第二引数をとることを表すアトムを用いて表現する。また、文節「金を」などは意味情報の値として、種類が金である駒を表す情報付き変数、*B~ (駒 (種類 金)) をとる。

4.2 Step2

形態素解析ルールの適用により入力文を文節ごとに分けると、各文節ごとに文法制約アトムと文構造制約アトムが生成される。

文法制約アトムは第一引数にある文節をとり、第二引数に全文をとる。第一引数の文節が第二引数の全文の中のどの文節と依存関係を持つか決定するアトムである。

```
(文法制約 *2~((単語  金) (付属語 を)
              (依存先  *22~(or 1 2 3 4 5 6 7))
              (意味 *B~(駒 (種類  金))))
         <全文>)
```

最初は第一引数の文節の依存先の候補はすべての文節となり、依存先情報の値として、or で始まりすべての文節番号をリストにもつ情報付き変数をとる。or で始まる情報付き変数は、その変数が or 以降の番号のいずれかに値が定まることを表している。解析が進むと、依存先の情報付き変数の or 情報のリストが短くなっていく。文法制約のアトムは依存先が完全に決定すると消去される。この例では、「金」が名詞であり、付属語が「を」であることから、動詞にしか係らないので、依存先が次のように変化する。

```
*22~(or 1 2 3 4 5 6 7) → *22~(or 3 7)
```

文構造制約アトムは、ルールの適用により、後方依存制約アトムと非交差制約アトムに展開される。後方依存制約とは、依存先が後方にあることを表現している。後方依存制約アトムは無条件に展開する事ができる。後ろから2番目の文節、「歩を」は後方依存制約より、係り先が最期の文節、「動かせ」に係ることが決定する。ここで、「金を」と「歩を」の両方が「動かせ」に係ることは無いので、「金を」の係り先は「取れる」に決定される。

もう一つの非交差制約とは、依存先を示す矢印が交わらないことを要請している。非交差制約アトムは第一引数に文節番号を、第二引数にその文節の依存先情報の値、第三引数に全文をもつ。

```
(非交差制約  1 *11~(or 2 4 6) <全文>)
(非交差制約  3 *33~(or 4 6) <全文>)
```

非交差制約アトムの存在により、この例文では、「先手の」が「飛車の」に係り、かつ「取れる」が「歩を」に係ることが無いと保証される。結果として文節の依存関係は図3のようになる。この例文は、構文解析だけでは表現が曖昧なため5通りの解釈が考えられる。

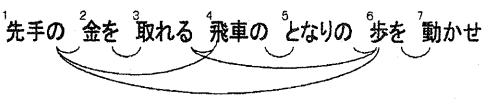

図 3: 文節の依存関係

4.3 Step3

文法制約アトムは依存先が決定されると消滅するが、名詞句が動詞と依存関係をとった場合、意味情報を制約するアトムを生成する。先の例文の場合、文法制約ルールの適用によって、「金を」がつぎの文節「取れる」に係ることからgetアトムが生成され、「飛車のとなりの歩を」からnextアトムが、「歩を」が「動かせ」に係ることからmoveアトムが生成される。

getは第一引数が第二引数を取れることを、nextは引数同士が隣り合っていることを、moveは第一引数の駒が第二引数の場所に動かせることをそれぞれ制約するアトムである。これらのアトムと文法制約のアトムは次のようになる。

```
(文法制約 *1~((単語  先手) (付属語 の)
              (依存先 *11~(or 2 4 6)
              (意味 *p~(指手 先手)))
          <全文>)
(文法制約 *3~((単語  取) (付属語 れる)
              (依存先 *33~(or 4 6)
              (意味 (get *A *B~(駒  (種類 金)))
          <全文>)
(get   *A   *B~(駒  (種類 金)))
(next *C~(駒  (種類 飛車)) *D~(駒 (種類 歩)))
(move  *D~(駒  (種類 歩)) *place)
```

構文解析のアトムと意味解析のアトムで互いに共有変数を持っている。getアトムなど構文解析で生成されたアトムが引数にもつ、駒を表す情報付き変数*B,*C,*Dは、それぞれの文節が持つ意味情報の値と単一化して共有変数となる。本システムでは共有変数として情報付き変数を用いる。情報付き変数は後ろのS式を参照したり、変形したりすることができる。あるルールの適用により情報付き変数の情報が変化すると、その情報が共通変数を持つ全てのアトムに伝わる。このようにして意味制約と構文制約との情報の授受が可能になる。

構文解析だけでは金を取れる駒がわからないため、現時点ではgetアトムの第一引数は不明である。従って「取れる」の依存先が不明である。歩を動かす場所が不明なため、moveアトムの第二引数は不明である。しかし、nextアトムだけは第一引数、第二引数ともに駒の種類が判明していて、*C、*Dが隣合った飛車と歩であることがわかる。よって構文解析がまだすべて終わっていないが、「飛車のとなりの歩を」の所を先に部分的に意味解析を行うことができる。このようなことができるのは、各解析を細かく制約処理別に定義し、処理手順を動的に決定しているためである。

意味解析では、盤面の情報(図4)と将棋の規則の知識などから、曖昧性のないようにどの駒でどの駒をとるのか、ということなどを決定する。図4の通りnextアトムの解析により*C、*Dはそれぞれ後手の6五の飛車と後手の7五の歩であることがわかる。飛車と歩を表す情報付き変数に情報が追加され、*C、*Dは次のように変化する。

```
*C~(駒 (種類 飛車))
     →  *C~(駒 (種類 飛車) (指手 後手) (場所 6五))
*D~(駒 (種類 歩))
     →  *D~(駒 (種類 歩) (指手 後手) (場所 7五))
```

後手

9	8	7	6	5	4	3	2	1	
香	桂	銀		王	金	銀		香	一
			金						二
歩	歩				歩	桂	歩	角	三
				金	飛				四
		歩	飛						五
		金				歩			六
歩	歩		歩	歩				歩	七
	角				銀				八
香	桂	銀		王			桂	香	九

先手

図 4: 盤面

ここで文節「先手の」を引数にもつ文法制約アトムに着目する。「先手の」は、構文解析により、駒に係ることはわかるが、3つの駒のうちどれに係るか(依存先が金か飛車か歩か)決定することができない。

```
(文法制約 *1~((単語  先手) (付属語 の)
              (依存先 *11~(or 2 4 6)
              (意味 *p~(指手 先手)))
          <全文>)
```

このアトムの第二引数の<全文>の中の*C、*Dも、先のnextアトムの展開により変化している。あらかじめ文節「先手の」には意味として指し手が先手であるという意味情報がつけられているのに対して、文節「飛車の」や、「歩を」の意味情報には指し手が後手であることが新たに加えられている。従って、これら二つとは依存関係は結び得ないので、「先手の」の依存先は残った

1つ、「金を」に絞られることになり、このアトムは展開される。

これが意味解析から構文解析への情報の伝達である。最初の構文解析では意味情報の制約による矛盾が生じなかった依存先の候補も、意味解析を行うことによって意味情報が伝搬され、それをもとに候補の絞り込みを行うことができる。結果的に、意味解析が構文解析の処理を促すことになる。

4.4 Step4

前節の解析結果により、金が先手の駒であることがわかった。残りのアトムは次のようになる。

```
(文法制約 *3~((単語  取れる) (付属語 ())
              (依存先 *33~(or 4 6)
              (意味 (get *A *B~(駒 (種類 金)(指手 先手))))
          <全文>)
(非交差制約  3 *33~(or 4 6) <全文>)
(get   *A  *B~((種類 金) (指手 先手)))
(move  *D~(駒  (種類 歩)(指手 後手) (場所 7五)) *place)
```

move のアトムは move ルールにより展開され、場所が決定される。この場合7五の歩が動けるのは7六しかない。あと判明していないのは、「取れる」が飛車と歩のどちらに係るか、ということである。どちらに係っても、非交差制約を満たしている。

ここで get アトムに着目する。盤面より、先手の金は二つあり、将棋規則ルールから、それらを取ることのできる駒はそれぞれ1つずつ存在することがわかる。

- 先手の (先手の7六の) 金を 取れる 後手の7五の歩
- 先手の (先手の5四の) 金を 取れる 後手の4四の飛車

これだけでは結局「取れる」が歩に係るのか飛車に係るのか決定できない。get アトムの解析で出た飛車、歩と、文に出てくる飛車、歩がそれぞれ同じものかどうか判定しなければならない。そのためには全文を引数に持つ、文法制約アトムも展開する必要がある。文に出てくる飛車、歩は、先程までの解析の結果によりそれぞれ後手の7五の歩、後手の6五の飛車であることが判明している。get アトムの飛車は4四の飛車なので異なるが、歩は同じ7五の歩なので、*A は後手の7五の歩であるとわかる。従って「取れる」は「歩を」と依存関係があることが分かり、以上より文全体の構造が明らかになったことになる。意味解析の結果、文は

- 先手の (7六の) 金を取れる (後手の6五の) 飛車のとなりの (後手の7五の) の歩を (7六に) 動かせ

となる。

4.5 ルール適用回数の比較

逐次処理と統合処理で以下の7つの例文を解析した。その時の等価変換ルールの適用回数と節の分岐数を表に示す。

1. 先手の飛車を取れる金を取れる歩を動かせ
2. 先手の7－6の飛車を取れる金を取れる歩を動かせ
3. 金を取れる歩を動かせ
4. 飛車のとなりの先手の金を取れる歩を動かせ
5. 先手の7－6の飛車のとなりの歩で取れる金を取れ
6. 先手の金を取れる飛車のとなりの歩を動かせ
7. 先手の7－6の飛車を取れる金を取れる駒を動かせ

処理		1	2	3	4	5	6	7
逐次	step	654	1016	309	645	839	996	926
	分岐数	2	5	16	2	10	6	5
統合	step	449	502	271	430	491	440	528
	分岐数	0	0	3	1	0	1	0

図 5: 解析結果の比較

以上より、構文解析と意味解析の統合処理を行なうことにより、等価変換ルールの適用回数と分岐数を減らすことができたことがわかる。

5 まとめ

本論文では、本研究で構築した等価変換による自然言語理解システムについて述べた。特に、解析空間内の円滑な情報伝搬機構を構築し、どのように各解析の相互作用が起こり得るかについて述べた。そして、等価変換に基づく方法で全解析の統合処理が可能であることを示した。実験結果として、構文解析と意味解析の相互作用により無駄な処理が省けることを示した。

参考文献

[1] 赤間清：項領域における包含制約の等価変換, 人工知能学会,1997

[2] 吹田慶子、赤間清、宮本衛市：未知単語を含む文の理解のための計算アーキテクチャ

[3] 高橋征義, 赤間清, 宮本衛市:等価変換に基づく構文解析と意味解釈の統合処理：電子情報通信学会 技術研究報告 ソフトウェアサイエンス研究会,No.5,(1995)

[4] C.S.Mellish：Computer Interpretation of Natural Language Descriptions,Ellis Horwood,1985.

[5] 畑山満美子, 赤間清, 宮本衛市：プログラム変換に基づくルールの追加による知識処理システムの改善法, 人工知能学会誌,1997