プログラム変換による自然言語理解の統合*

1G－1

高橋　征義　赤間　清　宮本　衛市†
北海道大学 工学部 情報工学科‡

1　はじめに

自然言語理解は、単語に関する知識や構文に関する知識、語用論的な知識、あるいは言語外的な知識を利用して行なわれる。

しかし、これらの知識は相互に依存しあっている。単語と構文的な知識だけで構文構造は決定できず、言語外的な知識を用いなければ文の意味は求まらない。そのため、これらを別々に処理しようとすれば、処理の効率が悪化する。そこで、これらの知識の統合、すなわち、さまざまな知識をまとめ合わせて利用することにより、処理の効率化を計ることが必要となる [5]。

本稿では、ルールに基づく宣言的プログラムのプログラム変換 [2] を用いて、自然言語理解の統合を行なう手法を提案する。宣言的プログラムのプログラム変換 [1] は、プログラムを確定節の集合で表し、プログラムをその宣言的意味を保存するように変換することにより計算を行なうものである。変換の規則はルールとして記述されるが、知識をプログラムと考えれば、それぞれの知識をいかに変形するかがルールに対応する。そして、ルールの適用はルール自体とは別に与えることができるので、あらかじめ具体的な実行の順序を与えられない場合にも、効率的な実行を行なうことができる。これは、対象領域に依存しない自然な統合の実現を可能にする。

2　必要な知識

自然言語理解の例として、対象世界を将棋の世界とする。ここでは、例えば人間が「歩を２－４に動かせ」という文と、将棋の盤面をコンピュータに与えると、コンピュータが「Move Fu at 2-3 to 2-4」といった命令を生成する、というシステムを考える。コンピュータは、与えた入力文を解釈し、実行が可能な命令に変換するのである。

このようなシステムを作るためには、さまざまな知識が必要となる。以下は、ここで用いている知識のおおまかな分類である。

- **入力文の受理に関する知識**: 入力文の各語と、その素性構造とを対応づける知識。
- **語の依存関係に関する知識**: どの語と語が依存関係 [6] にあるかについての知識。すなわち、語の係り受けの知識。
- **素性構造に関する知識**: 依存関係にある語同士、あるいは語自体の素性構造 [3] の知識。
- **将棋の規則に関する知識**: 駒の動かし方といった将棋の規則の知識。
- **将棋盤の盤面に関する知識**: その時点での盤面上の駒の配置の知識。

最初の３つの知識は自然言語理解一般についての、最後の２つの知識は将棋の世界に限定された知識である。対象世界が変わる場合は、最後の二つの知識を別の知識に置き変えればよい。

3　効率的な処理

同じ知識であっても、その用いられ方によって効率的に処理が行なわれる場合もそうでない場合もある。

例えば unfold 変換 [4] を行なう場合、変換の対象となる節からボディアトムを一つ選び、それと単一化できるヘッドを持つ節を用いて変換を実行する。この時、単一化できるヘッドを持つ節が複数ある場合、そのボディアトムを選んでしまうと、変換後のプログラムの節数は変換前に比べて増加する。

処理の効率化を行なうためには、「節の数をなるべく増加させない」ということが重要である。節数の増加

*Integration of Natural Language Understanding Using Program Translation

†TAKAHASI Masayosi, AKAMA Kiyosi, MIYAMOTO Eiichi
‡Hokkaido Univ.

は不必要な計算の増加につながりやすいので、どうしても行なわざるを得ない場合だけしか行なわないようにした方がよい。

節数の増加を起こさないようにすることは、制御の問題である。prologのように制御が固定されてしまっている場合は、節数を増やさないようにするために制御を変えることはできない。

ルールに基づくプログラム変換では、変換の規則はルールで与えられており、どのようなルールを優先的に適用するかは、ルールとは別に定めることができる。

ルールには、節数を増加させるルールも増加させないルールもありうる。これは、宣言的意味を保存するものなら、どのような変換でも許されるためである。できるだけ節数を増加させないようにするため、我々は節を増加させないルールを優先して適用するような制御を用いている。

4 実行結果

我々が現在試作している自然言語理解システムでの実行結果を以下に示す。

表 1は、このシステムに「歩を２－４に動かせ」という命令を与えた場合の、ルールの適用順序を示している。

表の数字は、一番左がルール全体の適用回数の累計、それ以外は種類別のルールの適用回数を表す。ルールはそれぞれ2節で述べた知識の使用に対応している。

表を見れば、基本的には左から右の順に処理が行なわれているのが分かる。これは、従来行なわれていた、辞書びきから始まって構文解析を経て意味解析へという処理の順序にほぼ沿っている、と考えることができる。

けれども、全体の処理は各段階ごとに別々に行なわれているのではない。ある一種類のルールの適用がまだ終わらないうちから、次のルール、あるいはさらにその次のルールの適用が行なわれている。これは、各々の処理が並行に行なわれていることを示している。

このように、節を増やさない実行を行なうために、適切なルールが用いられていることが分かる。

回数	入力	依存	素性	知識	盤面
50	50	0	0	0	0
100	22	17	11	0	0
150	2	12	36	0	0
200	2	25	23	0	0
250	2	26	22	0	0
300	2	37	11	0	0
350	0	2	48	0	0
400	10	0	8	30	2
450	0	2	34	10	4
500	0	3	47	0	0
550	0	0	37	5	8
600	0	0	0	19	31
601	0	0	0	1	0

表 1: ルールの適用順序

5 おわりに

プログラム変換による統合は、対象世界を限定するものではない。それぞれの対象世界をルールの形で記述すれば、構文に関するルールは変更を加えずに、統合することができる。

また、プログラム変換は、自然言語理解のみならず、明示的に知識を記述できるものならどんな課題にでも対応できる。実際、その他の分野への適用も試みられている。

参考文献

[1] 赤間清 (1993), 宣言型計算モデル, 情報処理学会、プログラミング研究会,12-10-PRG,pp.87-94.

[2] 赤間清 (1994), ルールを基礎としたプログラム変換,「自然言語処理における実働」シンポジウム論文集,pp.86-94.

[3] GUNJI,Takao(1987). Japanese Phrase Structure Grammar, D.Reidel,Dordrecht.

[4] TAMAKI,H. and SATO,T.(1984). Unfold/fold Translation of Logic Programs. Proc. of 2nd ILPC. pp.127-138.

[5] 橋田浩一 (1991), 自然言語処理における統合の諸相, コンピュータソフトウェア Vol.6 No.4,pp.3-16.

[6] 益岡 隆志：モダリティの文法、くろしお出版、(1991)