9B−2

自然言語インタフェースにおける制約ベース意味表現生成

山口 智治　市山 俊治
NEC 関西C&C研究所

1 はじめに

本報告では、自然言語インタフェースにおいて、入力された自然言語文からアプリケーションで実行するタスクを表す意味表現の生成過程、特に構造の決定手法について述べる。

意味表現をタスクを表すための言語であるとみなすと、意味表現生成とは言語生成であると言える。従来より、言語生成では構文構造木をトラバースする構造主導処理がよく用いられるが、この手法では自然言語表現の多様な構造に追従させることが困難であるという問題があった。

しかし、自然言語インタフェースにおいては、アプリケーションの機能によって概念の担う役割は限定され、多様な表現を入力とする場合にもこれらに対応して生成すべき構造は制限される。そこで、概念主導による構造生成を考える。まず自然言語文中の概念をアプリケーションで受理可能な記号に変換し、記号の担う機能から記号間に可能な構造を絞り込んで決定する。この手法を関係データベースを対象アプリケーションとして研究試作中の自然言語インタフェース[市山91, 谷91]に実装し、動作を確認した。

2 記述子間の結合による意味表現生成と結合の制約

2.1 意味記述子主導処理による意味表現生成

意味表現生成過程は、自然言語文の解析結果である概念表現を入力とし、アプリケーション言語を指向した意味表現を出力する。概念表現は自然言語文に現れた概念を表す概念記述子をノードとし、それらの格関係をリンクで表したネットワークであり(図1)、リンクによって構築された構造を概念構造と呼ぶ。意味表現はアプリケーションが受理する記号である意味記述子をノードとし、引数名をもつリンクで意味構造をなすネットワークの形で表される(図6)。

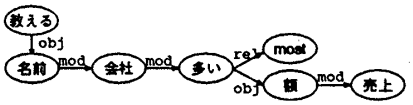

図 1: 概念表現の例

入力される概念表現に対し、まず概念記述子に対応する意味記述子を求める"記述子変換"を施し、次に"構造生成"を行なうという、2段階の処理を施して意味表現を生成する。記述子変換には[山口91]で述べた連想検索手法を用いている。

ネットワーク形式の意味表現は、ノード間をリンクによって結合することが生成の基本処理となる。この点に着目して、意味記述子のペアに基づいた意味構造生成をおこなう。記述子変換で求まった意味記述子の2つずつの組合せについて、その結合の可否を判定する。判定に基づいて、これら意味記述子の間をリンクでつなぐことによって構造を生成する。

Constraint-Based Generation of Task-Representation for a Natural Language Interface
Tomoharu YAMAGUCHI and Shunji ICHIYAMA
Kansai C&C Research Lab., NEC Corp.

判定は次に述べる制約を用いて、意味記述子のペアがこの制約を満足するか否かをチェックすることでおこなわれる。

2.2 結合の制約

意味記述子のペアについて、その結合の可否を判定するために対象アプリケーション言語の構文に関する**意味表現構文制約**、概念構造と意味の関係による**概念構造制約**、対象アプリケーションにおける意味記述子の解釈と機能の関係による**対象アプリケーション機能制約**の3種類の制約を用いる。

2.2.1 意味表現構文制約

意味表現上で意味記述子は引数をとる。同時に他の意味記述子の引数となり得る。これら引数のタイプを意味記述子が要求する引数タイプ、意味記述子が提供する引数タイプと呼ぶ。要求する引数には必須のものと任意のものがある。例えば、SQLを検索言語とするデータベースの検索にかかわる意味記述子が要求/提供する引数タイプを図2に示す。

提供する引数タイプ	意味記述子	引数名	要求する引数タイプ
cmd	select	object cond how	*nvexp* *cond* *how*
how	group	key cond	*field* *cond*
field	field	name cond table	フィールド名 *cond* テーブル名
value	value	value	値データ
nvexp	min	arg cond	*nvexp* *cond*
nvexp	max	arg cond	*nvexp* *cond*

注: 必須要求引数

図 2: 意味表現構文制約の一部

ある意味記述子のペアにおいて、一方の意味記述子が要求する引数のタイプを他方の意味記述子の提供する引数のタイプが充足可能であれば、その引数タイプについて、要求する意味記述子の引数として提供する意味記述子を結合するための意味表現構文制約を満足する。

ここで、充足可能とは、図3に示す引数のタイプの抽象化階層において、提供される引数のタイプが、要求されている引数タイプと一致するか、より下位にある状態を言う。

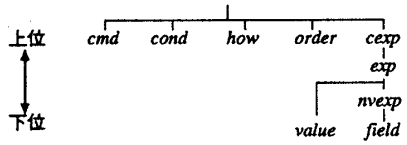

図 3: 引数タイプの抽象化階層

2.2.2 概念構造制約

意味構造上で結合される意味記述子のペアは概念表現上でもそれぞれの元の概念記述子間に特定の構造をとる場合が多い。ここでは、概念構造をトラバースして見つかる特定の構造から意味表現を生成するルールとしてではなく、特定の意

味記述子のペアについてそれらを結合可能にするための制約として概念構造を参照する。

また、多くの意味記述子は複数の引数を要求する。ある意味記述子が同じタイプの引数を複数要求するときにどちらの要求を充足するか、概念表現上の格ラベルを参照して決定する場合などでも概念構造制約は有効に働く。

2.2.3 対象アプリケーション機能制約

対象アプリケーションの機能のもつ特徴も意味記述子間の結合の判定に有効である。データベース検索の場合“検索言語中では、データはフィールドと対をなし比較演算子を介して結合される”という特徴をもつ。

このような特徴は対象アプリケーション機能制約となる。例えば、データの意味記述子と比較演算子の意味記述子のペアに対して、そのデータが格納されているであろうフィールドが求まらなければこれらの結合は許されない。この結合は一旦保留され、データと対になるフィールドと比較演算子の意味記述子ペアと合わせて両方の結合の制約が充足される。

また、データと対をなすフィールドや比較演算子を示す概念が概念構造中に存在するとは限らない。この場合にはこれらに対応する意味記述子も求まらないことになるが、対象アプリケーション制約を考慮することによって、省略された概念を補い、意味記述子を追加することが可能になる。

3 制約に基づく構造生成

3.1 結合テーブル作成

意味記述子が要求する引数のリストと提供する引数のリストを記した結合テーブルを用意する。図4は、会社名と売上額のデータがそれぞれ“会社”フィールドと“売上”フィールドに格納されているデータベースを対象として「売上額が最も多い会社の名前を教えて」という入力文によって作成される結合テーブルの例である(ただし、この時点では充足状況フィールドには何も記録されていない)。これらはそれぞれ、要求を満たす引数を提供する意味記述子、提供する引数で要求を満たす意味記述子を記録できる。

要求する引数の必須のものと任意のものをそれぞれ needs, wants と呼ぶ。また、提供する引数を supply と呼ぶ。

		ノードID	意味記述子	引数名	要求タイプ	充足状況
needs	1	3	max	arg	*nvexp*	supply 1
wants	1	0	select	object	*nvexp*	supply 3
	2	0	select	cond	*cond*	supply 2
	3	0	select	how	*how*	--
	4	3	max	cond	*cond*	--
	5	2	field(会社)	cond	*cond*	--
	6	5	field(売上)	cond	*cond*	--

		ノードID	提供タイプ	意味記述子	提供状況
supply	1	0	*cmd*	select	
	2	3	*nvexp*	max	wants 2
	3	2	*field*	field(会社)	wants 1
	4	5	*field*	field(売上)	needs 1

図 4: 結合テーブルの例

3.2 制約チェック、結合テーブル更新

needs/wants のリストから未充足の要求を持つ意味記述子を、supply のリストから提供が未充足の意味記述子をそれぞれから1つずつ選び出し、それらについて先に述べた制約のチェックによる結合判定を行なう。それぞれの制約の適用順序については図5の流れに従う。

まず、意味表現構文制約をチェックし、これを満足するペアのみについて他の制約をチェックする。概念構造制約を満足した意味記述子ペアの対象アプリケーション機能制約チェックや、対象アプリケーション機能制約チェックで保留された結合の条件となる他の意味記述子ペアの概念構造制約チェックなどのため、概念構造制約チェックと対象アプリケーション機能制約チェックは相互に働く。

各制約を満足して意味記述子のペアが結合可能と判定されたら、needs/wants のリストには要求を満たす引数を提供をする supply リストの意味記述子を、supply のリストには提供する引数で要求を満たす needs/wants リストの意味記述子をそれぞれ充足状況フィールドに記録する。

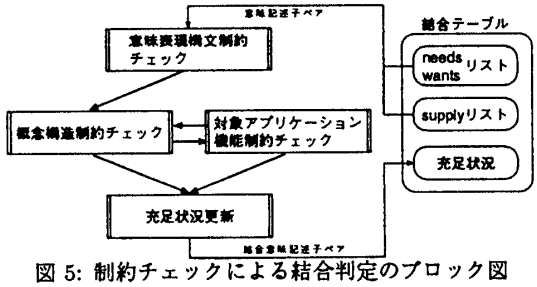

図 5: 制約チェックによる結合判定のブロック図

3.3 意味表現の生成

充足状況が記録された結合テーブルを参照して、各意味記述子をノードとし、それらの間に引数名をラベルとするリンクを張り、意味表現のネットワークを完成する。図6は図4の結合テーブルから生成される意味表現の例である。

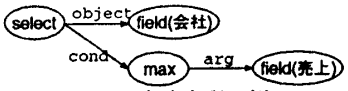

図 6: 意味表現の例

4 おわりに

ネットワーク形式の表現ではノード間をリンクによって結合することが構造生成の基本処理となることに着目して、意味記述子のペアに基づいた意味構造生成の枠組を提案した。記述子変換で求まった意味記述子の組合せについて、入力概念構造に基づく構造的制約、対象アプリケーションにおける機能的制約、意味表現の構文的制約の充足をチェックし、その結合の可否を判定する。判定に基づいてこれら意味記述子をノードとしてノード間をリンクでつなぐことによって構造を決定し、意味表現を生成する。

この枠組を関係データベースを対象アプリケーションとする自然言語インタフェースに実装、評価中である。

他のアプリケーションへの汎用性、アプリケーション毎に異なる意味表現構文や機能的制約を定義する手順の明確化、支援、自動化が今後の課題である。

参考文献

[市山 91] 市山俊治, 村木一至：自然言語インタフェースの構築キットの提案, 43 回情処全大 1H-2, 1991.

[谷 91] 谷幹也, 飯野香, 山口智治, 市山俊治：自然言語インタフェース構築キット:IF-Kit, 信学技法 NLC91-62, 1991.

[山口 91] 山口智治, 市山俊治：拡張意味ネットワークの連想検索に基づく言語理解, 43 回情処全大 5H-10, 1991.