構文解析を補助的に用いる意味解析

船越 孝太郎　中野 幹生　長谷川 雄二　辻野 広司
(株) ホンダ・リサーチ・インスティチュート・ジャパン
{funakoshi,nakano,yuji.hasegawa,tsujino}@jp.honda-ri.com

1　はじめに

自然言語処理を専門としない開発者でも，表現の多様性や文法の不適格性・未知語等に頑健な言語理解システムを容易に構築できるフレームワークが求められている．

入力表現からの意味表現の生成，すなわち意味解析は言語理解の中心となる機能である．意味解析の一つのアプローチは，キーワード集合や表現パターンによって抽出した情報で予め定義されたテンプレートの空欄を埋めるもので (例えば [3])，本稿ではこれをテンプレート方式とよぶ．テンプレート方式を用いれば言語理解システムの開発が容易になるが，複雑な要求を扱うことが難しくなる．

意味解析のもう一つのアプローチは，構文解析結果から再帰的な手続きによって意味表現を生成するもので，本稿ではこれを構文解析方式とよぶ．構文解析方式では複雑な要求表現を理解することができるが，システムの構築と保守が困難となる．また構築したシステムの頑健性は低くなりがちである．

言語理解システムを効率的に構築するアプローチとして意味文法 [5, 1] が提案されているが，このアプローチでは開発者が意味文法とドメインの両方に精通する必要がある [6]．

システムの頑健性を増すために構文解析方式とテンプレート方式を二段構えで用いるアプローチ [7] もあるが，このアプローチではキーワード抽出で対応できるような単純な表現に対してしか頑健性を向上させることができない．

テンプレート方式の弱点を克服するために，Knight ら [2] は語と語の関連づけルールを人手で作成している．しかしこのような関連づけルールの作成は単純なドメインでもかなりの労力が必要である．

本稿では，構文解析方式とテンプレート方式を融合した新しい意味解析手法を提案する．ドメインの知識を重視し，構文解析結果を補助的に用いることで，頑健かつ強力な言語理解システムとその迅速な開発を追求する．構文解析には既存の汎用構文解析器を用いるが，文法や語彙の追加修正など構文解析器自体の調整は不要である．従ってシステム開発者は開発を迅速化でき，また最新の構文解析器に容易に乗り換えられる．

提案する意味解析手法は，概念体系，意味フレーム，レキシコン，オノマスティコンの四種類のドメイン知識を用いる．表現パターンも文法も必要ないので，大量の類似した表現パターンの管理や副作用の見極めが困難な文法規則の保守から解放される．また [5, 7] とは異なり，構文木から意味表現を生成する規則を文法と別に用意する必要もない．

2　ドメイン知識

提案手法が用いる四種類のドメイン知識について説明する．

概念体系　いわゆるオントロジーである．図 1 は 5 節で実験に用いるホテル予約システム用の概念体系である．

レキシコン　各概念を言語的に参照するための語句の集合である．これらの語句を**概念表現**とよぶ．図 1 において括弧内に示されているものが各概念に対応する概念表現である．

オノマスティコン　各概念のインスタンスを言語的に参照するための固有表現の集合である．各インスタンス毎に，システムがインスタンスを識別するために用いる**インスタンスシンボル**と対にして固有表現を定義する．

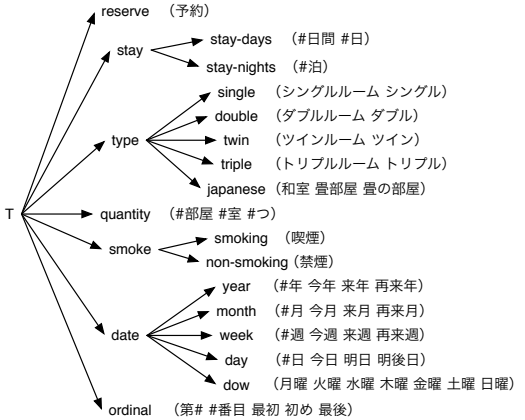

図 1: 概念体系

フレーム 1: `reserveF1`(*start*,*stay*,*type*,*quant*,*smoke*)
例: 「23 日から 3 泊で禁煙のシングルを 1 部屋予約」
フレーム 2: `reserveF2`(*start*,*end*,*type*,*quant*,*smoke*)
例: 「23 日から 26 日まで禁煙のシングルを 1 部屋予約」

スロット	対象概念	スロット指定	標示
start	`date`	“チェックイン”	“から”
end	`date`	“チェックアウト”	“まで”
stay	`stay`	“滞在日数”	—
type	`type`	“部屋の種類”	—
quant	`quantity`	“部屋数”	—
smoke	`smoke`	–	—

図 2: 概念 `reserve` のフレームとそのスロット

意味フレーム　意味フレームは各概念の属性集合として概念間の関係を定義し，意味表現の骨格となる．一つの概念に対して複数のフレームを定義することができ，これによって同じ概念を複数のパラメータの集合で表現できる．図 2 に，図 1 中の概念 `reserve` の二つの意味フレームとそれらのスロットの定義を示す．

各スロットの定義は，スロット名，スロット値の型，スロットを特定するための言語表現 (スロット指定)，同じくスロットを特定する標示からなる．スロット指定は，例えば「25 日チェックイン」といった表現のように，ある情報の意味役割を指定する．標示は「から」「まで」のような機能語のことで，スロット指定と同じ役割を持つが出現する場所が文法的に強く制限される．

3　意味表現

提案手法が出力する意味表現として**意味木**を定義する．意味木は，意味フレームの入れ子構造を木構造で表現したものである．意味木には 2 種類のノードがあり，どちらの種類のノードも他方の種類のノードをその子ノードして持つことができる．

1. 内容ノード
   内容ノードは，入力表現の中でそのノードに対応する部

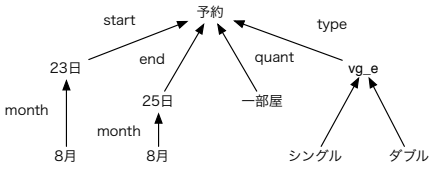

図 3: 例 (1) の意味木

分表現への参照とその部分表現が表す概念を保持する．また，そのノードがドメイン中のインスタンスを指し示す場合はインスタンスシンボルを，そうでない場合はその概念を規定するフレームを保持する．ある内容ノードの子ノードの最大数はノードが表すフレームの持つスロットの数である．子ノードはスロットの値を表現する．

2. 群ノード
群ノードは並列構造を表現し，列挙型と代替型の二つの型に分かれる．群ノードが持つ子ノードの数に制限は無い．群ノードは，並列構造の各要素の表層上での出現順序を子ノードの順序として保存する．

図 3 に例 (1) に対する意味木を示す．

(1) 8 月 23 日チェックイン 25 日チェックアウトでシングルとダブルを一部屋ずつ予約

4 意味解析

入力された自然言語表現から意味表現として意味木を生成する意味解析手法を提案する．提案手法は四段階からなり，以下の副節で各段階を順に説明する．

4.1 語句解釈

語句解釈は入力発話に対して正規表現によるパターンマッチングを行う．パターンマッチングは，ドメイン知識として与えられた概念表現，スロット指定，固有表現を対象とする．ある部分文字列に対して複数のパターンがマッチする可能性があるので，語句解釈は重なりなしで最大被覆をもたらすマッチング結果の集合を出力する．図 4 に例 (1) を語句解釈した結果を示す．

4.2 アクセス表生成

構文解析を行いアクセス表を生成する．アクセス表は構文解析結果 (複数候補があるときは 1 位のもの) を行列の形で表現したもので，構文解析木のノードである二つの語句の間にパスが存在するかどうかを表す．ここでは構文解析結果として依存構造を前提としているが，句構造から依存構造への変換は容易なので，句構造文法を使用する構文解析器を用いてもかまわない．構文解析器の出力を解釈してアクセス表に変換するプログラムさえ用意すれば，構文解析器には何を用いてもよい．図 5 に例 (1) を構文解析した結果を示す．本稿では CaboCha [4] を使用した．

汎用の構文解析器を調整無しで用いるのでドメインの語彙と構文解析結果 (の前提とする形態素列) が一致しない場合がある．そこで，構文解析結果を語句解釈の結果に摺り合わせてアクセス表を作成する．表 1 に，図 5 の解析木と語句解釈結果 (図 4) から生成したアクセス表を示す．表中の (1)〜(9) は図 4 中の語句解釈結果を参照している．構文解析結果とアクセス表を見比べれば分かるように，表中の x 行と y 列の交差点に記された数字は，x 行に示された語句が y 列に示された語句に到達するまでの距離 (木構造上のパス長) を表している．空欄箇所は 0 の意味であり，到達不可能であることを示している．例えば「ダブル」は「シングル」に距離 1 で到達できる．

「8 月 23 日」は CaboCha では一つの文節にまとめられてしまったが，図 1 の定義では,「8 月」と「23 日」は別々の概念になるので，表 1 では 2 行に分解されている．このように，複数の連続する概念が一つの依存構造解析単位としてまとめられてしまった場合は，解析単位の依存先をそのうちの最右概念の依存先とし，その他の概念は直右の概念に距離 1 で到達可能と見なす．逆に一つの概念が複数の依存構造解析単位に分割されてしまった場合は，最も右にある解析単位の依存先をその概念の依存先とする．それ以外の解析単位が別の概念に依存していてもそれらは無視する．複数の依存構造解析単位に分割された概念は，そのうちの任意の解析単位に依存する概念全ての依存先となる．

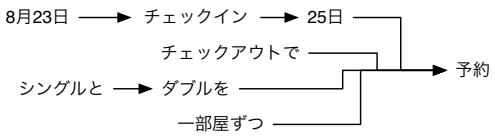

図 5: 例 (1) の構文解析結果

表 1: 例 (1) に対するアクセス表

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
(1)									
(2)	1								
(3)	2	1							
(4)	3	2	1						
(5)									
(6)									
(7)						1			
(8)									
(9)	4	3	2	1	1	2	1	1	

4.3 フレーム解釈

アクセス表を参考にして，語句解釈で抽出された各概念に該当するフレームとスロット割当を決定する．フレーム解釈はテンプレート方式言語理解の一種とみなせる．ただしそこに構文解析からくる制約を緩やかに導入している．

4.3.1 フレーム組み合せ

2 節で述べたように，各概念には複数のフレームを定義することができる．入力発話中のある概念表現に対応するフレームがどのフレームなのかはスロットの埋まり具合を評価しなければ決定できないので，事前に知ることはできない．そこでまず，各概念表現に対応するフレームの全ての組み合せを生成する．フレーム解釈のこれ以降の処理はフレーム組み合せ毎に行う．一つのフレーム組み合せから生成される意味表現は一つだけである．

4.3.2 入札

入力発話中に現れた概念は，通常，同じ発話内に現れた他の概念が持つフレームの，ある 1 スロットの値となる．各スロットの値をフレーム毎に要求する作業が入札である．アクセス表と同形のスロット割当表を用いて，各フレーム毎にスロット値への入札を行う．各フレームが入札できる箇所は，アクセス表の対応箇所に 1 以上の数値が記されている場所だけである．

表 2 は，表 1 に基づいて入札を行った結果である．ただし，スロット指定 (例 (1) 中の「チェックイン」,「チェックアウト」) は入札には関係しないので省略してある．表の各行がフレームとそのスロットの値を表現している．

3 行目は「23 日」と表現された概念 `day` のフレーム `dayF1` の入札内容を表す．`dayF1` は概念 `month` を値に取るスロット *month* を持つので，`month` の概念表現である「8 月」に入札している．8 行目は `reserve` の第 2 フレーム `reserveF2` の入札内容を表す．`reserveF2` はスロット値として共に `date` を取

8 月(1)23 日(2)チェックイン(3)25 日(4)チェックアウト(5) で シングル(6) と ダブル(7) を 一部屋(8) ずつ 予約(9)

(1)	概念: `month`	(2)	概念: `day`	(3)	スロット指定: *start*
(4)	概念: `day`	(5)	スロット指定: *end*	(6)	概念: `type`
(7)	概念: `type`	(8)	概念: `quantity`	(9)	概念: `reserve`

図 4: 例 (1) の語句解釈結果 (図 1, 図 2 の定義に基づく)

表 2: スロット割当表 (入札後)

	(1)	(2)	(4)	(6)	(7)	(8)	(9)
(1)							
(2)	*month*						
(4)							
(6)							
(7)							
(8)							
(9)		*start, end*	*start, end*	*type*	*type*	*quant*	

表 3: スロット衝突解決および文内省略解決後

	(1)	(2)	(4)	(6)	(7)	(8)	(9)
(1)							
(2)	*month*						
(4)	*month*						
(6)							
(7)							
(8)							
(9)		*start*	*end*	*type*	*type*	*quant*	

るスロット *start* とスロット *end* を持つので「23 日」と「25 日」に両者を入札している．

複数のフレームが一つの概念表現に入札しようとした場合は，依存構造上で最も近いもの (アクセス表で最小の値を持つもの) だけが入札できる．表 2 中のスロット割当表で「25 日」のフレームが「8 月」に入札していないのはこのためである．実際には「25 日」のフレームは「8 月」をスロットの値として取るべきであるが，これは後の文内省略解決で補償される．入札の段階ではこのような制限を設けておくことで誤った解釈 (妥当でない入札) を防ぐ．

4.3.3 スロット衝突解決

一つの概念表現 (項) は別の概念表現が表すフレームのただ一つスロットの値にしかなれない．従って，スロット割当表の各行において複数のスロットの値として入札が行われている箇所 (スロット衝突) は解消しなければならない．表 2 の場合，最終行の第 3 列と第 4 列がこれにあたる．

スロット衝突解決は以下の 3 つの基準に従って行う．

1. スロット指定
   入力言語表現中で，ある概念表現が特定のスロットの値であることを明示することがある．このような明示 (スロット指定) が存在した場合は，その明示されたスロットを採用する．例 (1) では,「チェックイン」が「23 日」を値とするスロット (*start*) を,「チェックアウト」が「25 日」を値とするスロット (*end*) をそれぞれ指定している．あるスロット指定がどの概念表現を修飾するのかは構文構造を無視した表層上の簡単なヒューリスティクスを用いて特定する．
2. 標示
   問題となっている概念表現をマークしている表現 (通常は格助詞などの機能語) があるかどうか入力発話に対して正規表現マッチングで調べる．もし標示が見つかれば，標示に対応するスロットを採用する．
3. スロット定義順序
   スロット指定もなく標示も見つからない場合，フレーム定義におけるスロット記述順にスロット値を分配する．語順の傾向をスロットの定義順としてコーディングしておく．

4.3.4 構文解析誤り回復

構文解析結果は必ずしも正確ではない．特に対象ドメインに特化されていない汎用解析器を用いた場合，文法的には誤りでないが意味的には誤った結果が出力されることが多くなってしまう．そのような誤りのうちのいくらかをここで回復するが，紙面の都合上詳細は省く．

4.3.5 文内省略解決

並列構造では頻繁に省略が起きる．並列構造の先頭の要素では言及された内容が，二番目以降の要素では省かれる現象である．以前の発話で指定されたためにおこる談話的な省略や，常識やドメイン知識に基づく省略と区別してこれを文内省略とよぶことにする．以下の二つの場合に，ある複数の表現が並列であるとみなす．A. ある概念表現のフレームの同一スロットの値である．B. それぞれが，ある概念表現のフレームの「起点・終点」の関係にあるスロットの値である．

A. の場合の例が，例 (1) の「シングル」と「ダブル」である．B. の場合の例が，例 (1) の「23 日」と「25 日」である．

ある概念表現 e がそれより左の別の概念表現 d と並列であると認められた場合，e に対応するフレームのスロットで値が空のものは，d に対応するフレームの同じスロットが入札している表現があれば，自身もそれに入札する．表 3 に，スロット衝突解決と文内省略解決を行った後のスロット割当表を示す．

4.3.6 スコア付け

以上でフレーム解釈処理は終了である．スロットの埋まり具合と文脈との適合度で，各フレーム組み合せにスコアを付ける．スコアは現在ヒューリスティクスを用いて与えているが，紙面の都合上詳細は省く．

4.4 意味木生成

フレーム解釈の結果から意味木を生成する．

4.4.1 スロット値グルーピング

まず最初に，複数のスロットの値からなる繰り返しのパターンをグループ化して分離する．例 (2) を考える．

(2) 明日の東京と明後日の大阪の天気

「明日」,「東京」,「明後日」,「大阪」の四つの概念表現は全て「天気」が表す概念のフレームのスロット値を表現している．しかし，この中で「明日」は「東京」だけに関係し，同様に「明後日」は「大阪」だけに関係する．

この表現に対して「明日の大阪」の天気を答えることは求められていない．単純に天気概念のフレームのスロット *date* が「明日」と「明後日」の二値を取ると考え，別のスロット *place* の値との関連を無視した意味表現にしてしまうとこのような間違いを犯してしまう．そこで上記のような複数のスロットからなる繰り返しパターンを検出し，グループ化しておく．

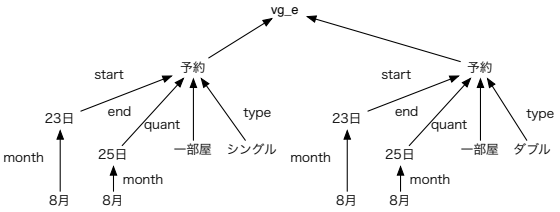

図 6: 図 3 中の意味木の因子分解結果

4.4.2 変換

スロット割当表を意味木に変換する．基本的には各概念表現／固有表現に対して一つの内容ノードを作るが，スロット割当が複数のスロット値グループに分離された概念表現に対しては，そのグループの数だけの内容ノードを作り，群ノードにまとめる．図 3 はこの段階のものである．

4.4.3 因子分解

図 3 は中間段階の意味木であり，これに対して因子分解を行う．因子分解された意味木はルートにただ一つの群ノードを持ち，それ以外のノードは全て内容ノードである状態になる．このような形に変形することで，意味解析結果を受け取るタスク処理モジュールの構築が容易になる．

特に指定のない場合，スロットの値は全て掛け合わせることによって組み合わせる．例えば「明日と明後日の東京と大阪の天気」という表現は,「明日の東京の天気」,「明日の大阪の天気」,「明後日の東京の天気」,「明後日の大阪の天気」の 4 つの表現に対応する意味木に因子分解される[1]．図 3 の意味木を因子分解した意味木を図 6 に示す．

5 対話実験と評価

提案手法を用いて，ドメイン知識を定義するだけで対話システムを構築できるフレームワークを実装した．フレームワークはドメイン非依存の対話管理機構を備えている．

このフレームワークを用いてホテル予約システムを構築した．意味解析に必要なドメイン知識は 169 行の XML ファイルに記述した．

9 人の被験者に事前に用意した五つの予約課題を行わせた．被験者は Web ブラウザ経由で予約システムとキーボード対話を行った．被験者は，予約システムが日本語を理解すること以外は何も知らされていない．図 7 に被験者と予約システムとの対話の抜粋を示す．

発話 U1 はシステムが理解できない「7 人」という語句を含んでいるが，提案手法が単純にそれを無視することでシステムは支障無く動作している．U3 は並列表現と文内省略を含んでいるが，それらは全て適切に処理されている．S2 はドメイン非依存の対話管理の結果としてシステムが生成した．S4 と S6 はドメイン依存の対話制御プログラムが生成した．対話制御プログラムについては本稿の範囲を超えるので説明を省く．

5.1 課題達成

課題達成率は 65.0%(26/40．システムの実装不具合で中断した 5 対話を除く) であった．この達成率自体は高いとはいえないが，図 7 に示すような複雑な理解を行うシステムを簡単な知識記述だけで実現できることを確認した．

課題不成功の主な原因の一つはユーザからのドメイン外の要求であった．課題不成功対話の半分は，システムがこれらの要求を無理やり理解しようとしたことが対話破綻の主たる原因と考えられる．

もう一つの大きな原因として，可算概念の扱いがある．課題不成功対話の残り半分のほとんどはこれが対話破綻の主たる原因と考えられる．被験者はしばしば「シングルを二つ予約して」のような要求の後に,「一つは禁煙でもう一つは喫煙」といった要求を行った．この場合，予約システムはシングル二部屋に対して単一の意味表現を割り当てるため，個別の部屋に別々の属性値を設定することができない．これを回避するためには，可算概念を個別に処理できるドメイン非依存の能力をシステムに備える必要がある．

U1: 8 月 15 日から 4 泊 7 人予約したい

S2: どんな部屋をご希望ですか? シングル/ダブル/ツイン/トリプル, 及び和室 (4 人部屋) がございます.

U3: 喫煙のシングル 1 部屋，禁煙の和室，ツイン一部屋

S4: 以下の内容でよろしいでしょうか？
項目 1: チェックイン:2006/8/15, チェックアウト:2006/8/19 (4 泊) 部屋種:シングルルーム 部屋数:1 喫煙
項目 2: チェックイン:2006/8/15, チェックアウト:2006/8/19 (4 泊) 部屋種:和室 部屋数:1 禁煙
項目 3: チェックイン:2006/8/15, チェックアウト:2006/8/19 (4 泊) 部屋種:ツインルーム 部屋数:1 禁煙

U5: はい

S6: 承知しました．料金は，108000 円です.

図 7: 実験で観察された対話の一例

5.2 意味解析

実験で得られた 372 発話からドメイン外の要求や一つだけ概念を含む発話を除くと 151 発話が残る．この 151 発話について提案手法の性能を評価した．

提案手法はこの 151 発話中に，フレームとそのスロットの値の関係にある二つの概念表現の組を 453 組認識した．94.0% (426 組) は正解であった．このことから提案手法がドメイン内の要求についてはその中に含まれるほとんどの概念を正しく理解したと言える．これらの 426 組のうち，221 組だけが対応する依存構造の上でも適切な依存関係を持っていた．従って，もし従来の様に構文解析結果に強く依存した意味解析を行った場合，48.8% (221/453) だけが正しく理解できたことになる．

参考文献

[1] John Dowding, Jean Mark Gawron, Doug Appelt, John Bear, Lynn Cherny, Robert Moore, and Douglas Moran. Gemini: A natural language system for spoken-language processing. In *Proceedings of the 31st Annual Meeting of ACL*, pp. 54–61, 1993.

[2] Sylvia Knight, Genevieve Gorrell, Manny Rayner, David Milward, Rob Koeling, and Ian Lewin. Comparing grammar-based and robust approaches to speech understanding: a case study. In *Proceedings of EUROSPEECH 2001*, pp. 1779–1782, 2001.

[3] Takashi Konashi, Motoyuki Suzuki, Akinori Ito, and Shozo Makino. A spoken dialog system based on automatic grammar generation and template-based weighting for autonomous mobile robots. In *Proceedings of INTERSPEECH 2004*, pp. 189–192, 2004.

[4] Taku Kudo and Yuji Matsumoto. Japanese dependency analysis using cascaded chunking. In *CoNLL 2002: Proceedings of the 6th Conference on Natural Language Learning 2002 (COLING 2002 Post-Conference Workshops)*, pp. 63–69, 2002.

[5] Stephanie Seneff. TINA: A natural language system for spoken language applications. *Computational Linguistics*, Vol. 18, No. 1, pp. 61–86, 1992.

[6] Ye-Yi Wang and Alex Acero. Rapid development of spoken language understanding grammars. *Speech Communication*, Vol. 48, No. 3–4, pp. 390–416, 2006.

[7] 山本幹雄, 伊藤敏彦, 肥田野勝, 中川聖一. 人間の理解手法を用いたロバストな音声対話システム. 情報処理学会論文誌, Vol. 36, No. 4, pp. 471–482, 1995.

[1] 例 (2) ではスロット値グルーピングによって「明日の東京の天気」と「明後日の大阪の天気」に分離されることに注意.