論　文　　Technical Paper

事象駆動型と予測駆動型を融合した文脈解析手法

The Integration of Event-driven and Expectation-driven Text Understanding within Parsing

上原　邦昭*　垣内　隆志*　豊田　順一*
Kuniaki Uehara　Takashi Kakiuchi　Jun-ichi Toyoda

*　大阪大学産業科学研究所
The Institute of Scientific and Industrial Research, Osaka Univ., Osaka 567, Japan.

1986年4月26日　受理

Keywords: Prolog, text understanding, actor, coherent relation, common sense knowledge.

Summary

This paper describes an integrated parser for text understanding. In the integrated parser, processes of event-driven and expectation-driven understanding occur as an integral part of the parsing process. That is, mechanisms of event assimilation, event search, and 'top-down' inference are incorporated into a single module. The mechanisms of event assimilation and event search are used to find coherent relations between successive portions of a text (i.e., event-driven understanding). The 'top-down' inference mechanism using common sense knowledge is utilized to expect what is coming next (i.e., expectation-driven understanding).

The common sense knowledge for expectation-driven understanding, called hierarchical event structure, is a property inheritance net that organizes a hierarchy of events. The hierarchical event structure is encoded explicitly into computational entities called actors which were proposed by Hewitt. When the integrated parser inputs the word associated with a particular event in the hierarchical event structure, it sends a message to the corresponding actor. The actor expects what kind of events are likely to occur in the text. Events extracted from the text are stored in an actor 'EVENT'.

The knowledge for event-driven understanding, which is associated with lexical entries, is utilized to detect the coherent relation between events. Each lexical entry is also encoded into an actor, whereas the knowledge for event-driven understanding is encoded implicitly within the procedural information of the actor. When the actor 'EVENT' receives the message followed by the procedural information, it extracts the coherent relation between the stored events and the input sentence. The contextual information contained in the text is, thus, available to resolve anaphora and to select meanings of ambiguous words during the parsing of a sentence.

1. はじめに

計算機によって自然言語を理解するためには，構文・意味情報だけでなく，文脈情報も考慮しないと厳密な解析が行えないことが指摘されている．このような状況から，最近の自然言語理解システムでは文脈解析まで含んだものが増えてきている．文献(5)によると，文脈は下記のように定義されている．

(1) 広義：状況と同じ意味に用いられ，その環境や場面などの条件を表す非言語的なものを含む．

(2) 狭義（言語的）：①文章の中での文と文の続きぐあい．②文中での意味の続きぐあい．

従来より提案されている自然言語理解システムは，文脈情報の利用の仕方に基づいて2つのグループに分類することができる．1つは，Schankらの SAM[10]，

FRUMP[2]などのような予測駆動型システムである．これは，次にどんな事象（以降では，特に断らない限りイベントと呼ぶ）が起こりうるかについて，トップダウン的に予測しながら文章を理解するものである．予測駆動型システムでは，あらかじめ用意された常識的知識を用いて解析を進めるために，(1)の意味における文脈を取り扱うことができる．しかしながら，前もって細部にわたるすべてのイベントの可能性を列挙しておかなければならないという問題がある．もう一方のグループに当てはまるものは，事象駆動型システムである．事象駆動型システムは，文章中に含まれるイベントが，一種の連結性を保っている場合にのみ，文章は意味をなしているという仮定に基づいたものである．このような立場に立つものに，Hobbs の coherence[4]，Schank の causal chain[10]などがある．事象駆動型システムでは，文と文の連結性をもとに，部分から全体へと文章をボトムアップ的に解析するために，(2)の①の意味における文脈を取り扱うことができる．しかしながら，文章全体が表している一連の状況や場面の流れといったものを把握できないという問題がある．このように，文章はどちらか一方の解析手法のみで理解できるわけではなく，両者を融合して互いの欠点を補いながら解析を進める必要がある．

一方，従来の自然言語理解システムでは，構文・意味解析から文脈解析へと一方向的に進行するものが多かった．たとえば，BORIS[8]は，開発当初，文章の意味解析と文脈解析をそれぞれ Conceptual Analyzer (CA) と Event Assimilator (EA) からなる2つのモジュールに分けていた．このため，CA は入力文の意味解析時に EA から得られる文脈情報を有効に利用できないという問題があった．また，CA の抽出する詳細な意味情報を EA の予測に反映させることができないという問題があった．

本報告で提案する自然言語理解システム IP (Integrated Parser)[12]では，予測駆動型と事象駆動型を融合した文脈解析メカニズムを導入し，(1)および(2)の①の意味における文脈を取り扱っている．また，構文・意味解析中に文脈解析メカニズムを起動し，文脈解析で得られた情報を構文・意味解析にフィードバックさせることができるために，効率的かつ人間の言語理解過程に近い形で解析を進めることが可能になっている．さらに，システムの制御構造をアクター理論[3]に基づくメッセージ交信制御に統一し，システム全体のモジュラリティを高めている．IP のデータ構造には LFG (Lexical Functional Grammar)[1]の機能構造 (functional structure) を拡張したものを用いており，文の意味表現と文脈情報を統一的な内部表現形式で記述できるようになっている．

2．IP の文脈解析

2・1　事象駆動型解析に必要な知識

文章中に現れるイベントは，一種の連結性を保っていると考えられる．イベントの連結性とは，あるイベントの生起が，文章中の他のイベントの生起と何らかの関連を持つことを意味している．たとえば，あるイベントの生起が原因となって，他のイベントが生起する原因・結果関係や，あるイベントの生起が，他のイベントの生起を可能にする可能化関係などは，イベントの連結性を保つのに役立っている．このように，2つのイベントに連結性があるとき，両者間で成り立つ関係を連接関係 (coherent relation) と呼ぶ[4]．連接関係にある文の並びは，出来事の説明文や順序立てて物事を表現する場合などの談話中においてみられる．本節では IP の文脈解析における連接関係の抽出について述べる．

IP では，イベントを，pred(arg 1, arg 2, …) という述語形式で記述している．pred は，動作（アクションと呼ぶ）あるいは状態（ステートと呼ぶ）を表している．また，arg 1, arg 2, …は，イベントの動作主や動作対象などを示している．たとえば，文 “John gave Mary a toy.” が表すイベントは，$give('John', 'Mary', 'a toy') と記述される．これは LFG での意味述語 (semantic form) に相当している．ただし，引用符 (') で囲まれた各語句は，対応する部分機能構造を示しているが，本報告では簡単のために上例のように表すものとする．

IP が取り扱う連接関係には以下の5種類がある．

(1)　アクションが生起した結果，新たなステートが生起する (result)．
　例：食事をする→ (result) →満腹になる．

(2)　ステートが生起しているため，他のアクションが生起可能になる (enable)．
　例：駅にいる→ (enable) →列車に乗る．

(3)　ステートの生起が，新たなアクションの生起理由になる (reason)．
　例：空腹である→ (reason) →食事する．

(4)　アクションの生起が，他のアクションを生起可能にする (enable)．
　例：乗る→ (enable) →降りる．

(5)　ステートの生起が新たなステートの生起原因になる (cause)．

例：けがをする→(cause)→死ぬ.

これらの連接関係を抽出するには，前向きに「現在のイベントから次に生じうるイベントを予測する」，または後ろ向きに「現在のイベントの原因となったイベントを探し求める」という2種類の推論が必要である.

この種の推論を行うために，前提(presupposition)と帰結(prediction)という概念を導入する．あるイベントが生ずるために，すでに生起していなければならないイベントを前提と呼ぶ．また，あるイベントが生起した結果，新たに生起するだろうと予測できるイベントを帰結と呼ぶ．連続した2つのイベント，E0, E1 (E0, E1という時間の流れでイベントが生起するものとする)の間の連接関係は，E0の帰結とE1, E0とE1の前提，およびE0の帰結とE1の前提の3種類のマッチングによって抽出される(Fig. 1参照).

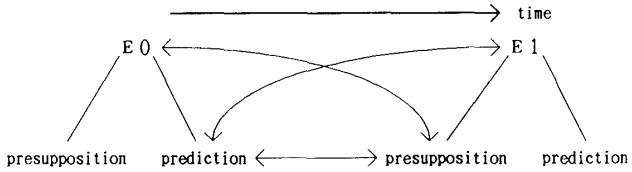

Fig. 1 Matching between events E0 and E1.

イベント間のマッチングは以下のように行われる.

(1) 2つのイベントの述語名が，同一であるかどうかをチェックする.

(2) イベントの引数間でマッチングを行う．すでに述べたように，イベントの各引数には機能構造が代入されているので，実際のマッチングは機能構造間で行われる．機能構造どうしのマッチングアルゴリズムについては前稿[12]に示している.

2・2 予測駆動型解析に必要な知識

事象駆動型の文脈解析では，イベントの連結性のみを対象としているために，以下のような文章から連接関係を抽出することはできない．この文章は，朝の場面に固有なイベントの流れを表したものである.

① Taro woke up at 8 in the morning.

② He washed his face and ate breakfast.

各文は，「起きる」，「顔を洗う」，「食事をする」という一連のアクションを表している．事象駆動型では，「起きる」というアクションから「顔を洗う」というアクションを予測できず，両者の関係を抽出することはできない．同様に，「顔を洗う」というアクションから「食事をする」というアクションを予測できないために，両者の関係を抽出することはできない．これは，場面に特有な情報，あるいは常識といった知識を事象駆動型で取り扱うことができないためである.

IPの文脈解析では，上例のような場面に固有なイベントのつながりを抽出するために，常識的知識を用いた予測駆動型の文脈解析を行っている．この常識的知識として，イベントの順序関係を表す枠組みをあらかじめ用意している．たとえば，「強盗が起こった」，「捜査が行われた」，「犯人が逮捕された」というイベントの順序関係はFig. 2のように表される．図中

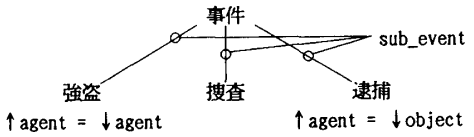

Fig. 2 Hierarchical event structure with 'sub_event' links.

のsub_eventというリンク名は，上位のイベントが下位のイベント系列から構成されていることを表している．しかしながら，この枠組みのみでは「殺人」に関する文章を理解する場合にも，「殺人」，「捜査」，「逮捕」というイベント系列を表すための別の枠組みが必要になる．このように，類似のイベント系列に対してそれぞれ固有の枠組みを用意していたのでは，莫大な記憶容量を要し非効率的である.

この問題を解決するために，「強盗」，「殺人」といったイベントの上位に位置するイベントとして「犯罪」を用意し，Fig. 3のようにイベント間の包含関係を階層的に構成する．ただし，is_aリンクは，下位のイベントが上位のイベントの一例であることを表したものである.

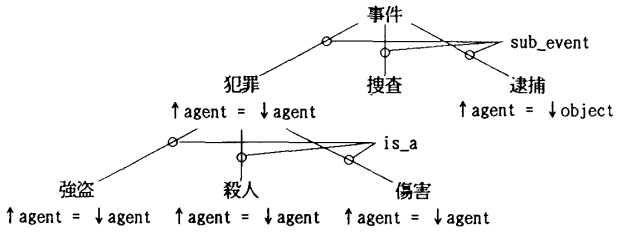

Fig. 3 Hierarchical event structure augmented with 'is_a' links.

上で述べた常識的知識の枠組みを階層的事象構造(hierarchical event structure)と呼ぶ．階層的事象構造は下記の形式で記述される.

事件；sub_event：犯罪(↑agent=↓agent),
捜査,
逮捕(↑agent=↓object).

犯罪；is_a：強盗(↑agent=↓agent),
殺人(↑agent=↓agent),
傷害(↑agent=↓agent).

この階層的事象構造は，Fig. 3のイベントの流れを表したものである．「逮捕」に付随する式↑agent=↓objectはスキーマと呼ばれ，イベントの行為者(agent)や対象物(object)などに対する制約条件と

して働いている．また，メタ変数↑は，上位のイベント「事件」を，メタ変数↓は，自分自身のイベント「逮捕」を表している．したがって，このスキーマは，「ある事件における強盗の行為者は逮捕の対象者と同一である」ということを表していることになる．

2・3　事象駆動型と予測駆動型の融合による文脈解析

本項では，事象駆動型解析と予測駆動型解析の融合について述べる．構文・意味解析過程で，連接関係を抽出するための特定の動詞が発見されると，事象駆動型解析が起動される．また，特定の場面（イベント）の連想キーとなる語句（動詞，あるいは名詞）が発見されると，予測駆動型解析が起動される．以下では，例文を用いてIPの文脈解析過程を具体的に説明する．

まず，事象駆動型解析について考える．

文①　Taro picked up the ball.

文②　He threw it.

文①からは，イベント $pick_up('Taro', 'the ball') と帰結 $have('Taro', 'the ball') が抽出される．文②の構文・意味解析過程で動詞 'threw' を発見すると，事象駆動型解析が起動される．動詞 'threw' を発見した段階では，まだ目的語 'it' を解析していないために，不完全なイベント $throw ('he', ?) しか抽出できない．

一方，動作主が「物を投げる」には，すでに「その物を所有していなければならない」ために，文②の前提は $have ('he', ?) となる．したがって，文①の帰結と文②の前提をマッチングすることができ，原因・結果関係 (result) が成立することがわかる．

文章中では，同一語句の反復生起を避けるために，照応表現が用いられることがあるが，代名詞の照応先は，イベント間の連接関係の抽出と同時に決定することが可能である．たとえば，文②の 'he' が文①の 'Taro' を指していること，および動詞 'threw' の目的語 (?で表した引数) には名詞句 'the ball' と適合可能な名詞句のみが予測されるために，'it' が文①の 'the ball' を指していることがわかる．Fig. 4 に抽出された連接関係を示す．

次に予測駆動型解析について考える．

文③　The man committed murder.

文④　He was arrested.

文③からは，イベント $commit('the man', 'murder') が抽出される．次に $commit の第2引数 'murder' からイベントが「殺人」を表していることがわかり，Fig. 3 の階層的事象構造を用いて，後続の文では「捜査」，続いて「逮捕」に関することが言及されているだろうと予測できる (Fig. 5 参照)．

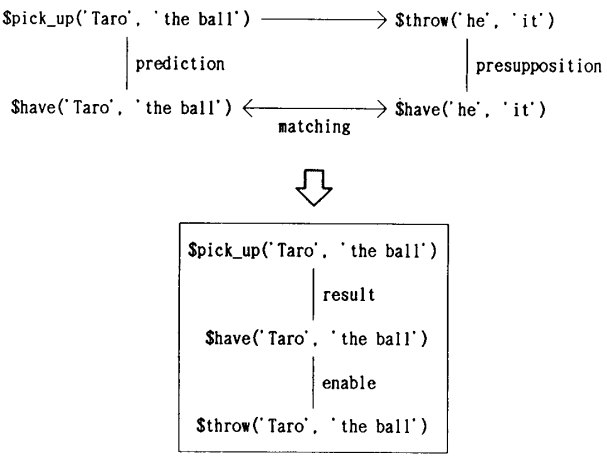

Fig. 4 Extraction of coherent relations between two events.

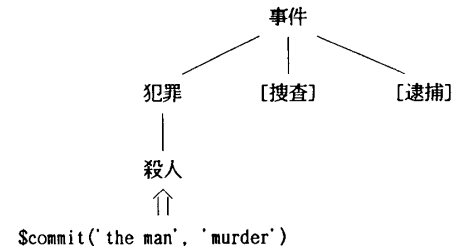

Fig. 5 Expectation of events by using the hierarchical event structure.

続いて文④を解析すると，動詞 'arrest' から「人が逮捕される」場面を表していると判断され，予測駆動型解析が起動される．予測駆動型解析では，文③の予測と文④から抽出されるイベント $arrest(?, 'he') が一致するかどうか判定される．この結果，予測「逮捕」と文④が一致するために，「殺人」の場面に固有なイベントのつながりを発見することができる．

また，予測駆動型解析では，各イベントに付随するスキーマを評価し，照応指示を解決できるようになっている．たとえば，上例では「ある事件における強盗の行為者は，逮捕の対象者と同一である」という制約条件を満足しているために，文④の 'he' が文③の 'the man' を指していることがわかる．

予測駆動型解析を起動するための，トリガとなる連想キーは，特定の単語とあらかじめ対応づけて用意している．上例では，動詞 'arrest' が「逮捕」の場面を表すイベントの連想キーとしての役割を担っている．しかしながら，Schank らも指摘しているように[10]，特定の単語を単純に連想キーとして用いた場合，階層的事象構造中の適切なイベントを呼び出すことができない場合がある．このような場合には，特定の単語だけでなく，文全体の意味やその文の置かれた文脈を考

慮する必要がある．たとえば，以下の文⑤，文⑥を解析する場合，

文⑤　He committed a theft.

文⑥　He committed a robbery.

文⑦　He was accused of theft.

動詞 'commit' は，それ自体で特定の場面を表すことができないために，イベントの連想キーにすることはできない．また，文⑤，文⑦は，名詞 'theft' を単純にイベントの連想キーとすることはできないことを示している．このような問題を解決するために，IP では動詞と格要素の両方の情報を用いて連想キーとしている．たとえば，動詞 'commit' の辞書中には「目的格に現れる語句を連想キーとして用いる」ことを記述し，文⑤，文⑥の問題を解決している．また，動詞 'accuse' は「告発」の場面に対応する連想キーとしており，文⑦が「盗み」の場面として解釈されることはないようにしている．

逆に連想キーから複数の候補が得られる場合には，文脈情報を利用して候補の絞り込みを行っている．たとえば，動詞 'shoot' は，イベント「殺人」，「傷害」，「射撃」などの連想キーとなる可能性がある．このような場合，すべての候補からそれぞれ予測を生成すると，多数の予測が得られることになり，後続の文から抽出されるイベントとのマッチングで組合せの爆発が生じる可能性がある．この問題を解決するために，IP では複数の候補が得られた場合に，予測を遅延 (freeze) している．遅延された予測は，後続のイベントから得られる文脈情報を利用して候補の絞り込みを行った後に，再実行 (force) される．たとえば，イベント $shoot に続いて $is_injured が抽出されると，事象駆動型解析で以下の連接関係，

$shoot→ (result) →$ is_injured

が発見され，「傷害」のみが有効となる．この結果，遅延されていた予測が再実行され，さらに「捜査」，「逮捕」が予測される．遅延評価の具体的なメカニズムについては，前稿 [12] で詳細に述べている．

2・4　Intersection search の導入

新聞記事のような文章では，まず大まかな枠組みを述べ，その後詳細な部分について言及するという形で話が展開することが多い．たとえば，以下の文章，

'The robbery occurred at around 9:40 p.m. Sunday. The robber threatened a young man with a handgun-like weapon. ...'

では，まず「強盗」について記述され，次いで「脅し」の状況が述べられている．これは，階層的事象構造を上位のイベント「強盗」から，下位のイベント「脅し」へたどることに相当している (Fig. 8 参照)．2・2 項で述べた予測駆動型解析では，階層的事象構造を上位に向かってたどるのみで，下位のイベントの探索については考慮していない．したがって，上例のような文章からイベント間のつながりを適切に抽出することはできない．このような問題を解決するために，Quillian の提唱したセマンティックネットワークにおける intersection search [9] の概念を用いて予測駆動型解析を拡張する．

intersection search は，後続の文から抽出されるイベントが，階層的事象構造から得られる予測とマッチングできない場合に起動される．これは，前文から抽出されるイベントと後続の文から抽出されるイベントが，階層的事象構造中で，リンクがつながっているかどうかを調べる操作にあたる．たとえば，上例では第 1 文のイベントから「捜査」，「逮捕」が予測されるのみで，第 2 文から抽出される「脅し」とマッチングできない．このような場合，「強盗」および「脅し」から互いに階層的事象構造のリンクをたどり，両者が共通な上位のイベントを持つかどうかを調べる．上例では，「強盗」自身がその共通なイベントにあたり，互いに意味的なつながりを持っていることがわかる．

3．アクターによる実現

英語では，文の表すアクションやステートは主に動詞によって決定されるので，対応するイベントは，それぞれの動詞辞書中にスキーマとして記述している．たとえば，動詞 'eat' の辞書規則は以下のように記述される．

```
eat ; v : [↑obj concept=ᶜfood,                          (1)
          ↑event=$eat(↑subj, ↑obj),                     (2)
          ↑presupposition=[(reason,
              $is_hungry(↑subj))],                      (3)
          ↑prediction=[(result,
              $is_satisfied(↑subj))]],                  (4)
        [↑obj concept=ᶜmetal,                           (5)
          ↑(↓pcase)=ᶜinto,                              (6)
          ↑event=$corrode(↑subj, ↑obj),
          ↑prediction=[(result,
              $is_corroded(↑subj))]].
```

上の辞書規則は，(1) 'eat' の目的語が「食物」であれば，「食べる」の意味を，また，(5)目的語が「金属」であり，かつ(6)動詞の後に前置詞 'into' を伴うならば「腐食する」の意味を採ることを示している．さらに，前提(3)には，「食事をする」理由として主格が「空

腹である」こと，また帰結(4)には主格が「満腹になるだろう」ということが記述されている．

前稿[12]でも述べたように，IP の辞書規則は，Intermission[6] の形式に従った Prolog プログラムに変換される．Intermission は，Prolog 上でアクターの概念を実現したもので，アクター間での制御の受渡しは，パターンマッチングによるメッセージ交信のみで行われる．たとえば，イベント(2)，前提(3)，帰結(4)は以下の Prolog プログラムで表現されるアクター $eat に変換される．

```
$eat(process, F) :-
    event(process, [F, $eat(F.subj, F.obj),
        [(reason, $is_hungry(F.subj))],
        [(result,$is_satisfied(F.subj))]]).
```

変数Fは，辞書規則中のメタ変数↑と対応している．メタ変数↑は，構文・意味解析過程で抽出される文の機能構造を指しており，イベントと文の機能構造を結び付けるために用いている．

辞書規則 'eat' に対応するアクター（辞書アクターと呼ぶ）が，「文脈解析を実行せよ」という process メッセージをアクター $eat に送信すると，アクター $eat はアクター event にイベント・前提・帰結を process メッセージとともに送信する．process メッセージを受け取ったアクター event は，すでに内部状態として蓄えられているイベントを1個ずつ取り出し，前提 $is_hungry(F.subj) あるいはイベント $eat (F.subj, F.obj) とマッチングを試みる．いずれかとマッチングが成功して連接関係が抽出されると，両者の連接関係を表す機能構造が生成される．さらに，現在解析中のイベントおよび帰結を内部状態として蓄える．これら一連の動作が2・1項で述べた事象駆動型解析に相当している．Fig. 6 は，Fig. 4 の連接関係を機能構造で表現したものである．

予測駆動型解析で用いられる，階層的事象構造の各イベントも，辞書アクターと同様に，Intermission

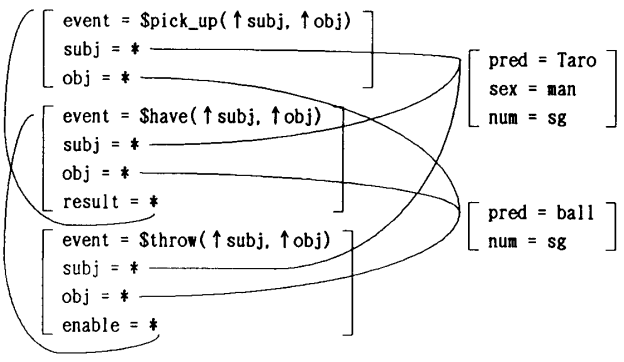

Fig. 6 Functional structure representation of coherent relations.

形式の Prolog プログラムに変換される．アクターとして実現された階層的事象構造は，expect メッセージを受信すると，上位のアクターを能動的に探索し，新たに生起しうるイベント系列を予測する．後続する文が予測のうちいずれかと一致する場合は，文章のつながりが発見される．2・2項で述べた階層的事象構造のうち，イベント「殺人」および「犯罪」に対応するアクターは以下のように変換される．

```
殺人(expect, F3) :-
        犯罪(expect, F1),                    (1)
        殺人(new, F3),                       (6)
        f-st(define, F3.is_a = F1),
        f-st(define, F1.agent = F3.agent).
殺人(new, F) :-
        f-st(new, F),
        f-st(define, F.self = 殺人).
犯罪(expect, F1) :-
        犯罪(new, F1),                       (2)
        事件(new, F2),
        f-st(define, F1.sub_event = F2),     (3)
        f-st(define, F1.agent=F2.agent),     (4)
        event(new, [F2, [捜査, 逮捕]]).      (5)
犯罪(new, F) :-
        f-st(new, F),
        f-st(define, F.self = 犯罪).
```

2・3項の例（文③と文④）を用いて上のプログラムの動作を説明する．ただし，以下の説明中の番号は，プログラムの文番号，および Fig. 7 の各番号とそれぞれ対応している．まず文③を解析するとアクター「殺人」が起動される．詳細については述べないが，階層的事象構造に対応するアクターは以下の手順で起動される．

階層的事象構造中の各イベント名は，それぞれ該当する辞書アクターにリスト（連想キーリストと呼ぶ）として登録されている．構文・意味解析過程で辞書アクターに起動がかけられると，辞書アクターは，連想キーリストに登録されたアクターに「予測を生成せよ」という expect メッセージを送信する．

(1) expect メッセージを受信したアクター「殺人」は，上位のアクター「犯罪」しか知らないために，予測を生成できない．このため，さらに上位のアクター「犯罪」に expect メッセージを送信する．

(2) メッセージを受信したアクター「犯罪」は，自分自身と上位のアクター「事件」に new メッセージを送信して，それぞれの階層的事象構造のインスタンス F1, F2 を生成する．ここでインスタンスとは，スキーマを評価して具体的にメタ変数の

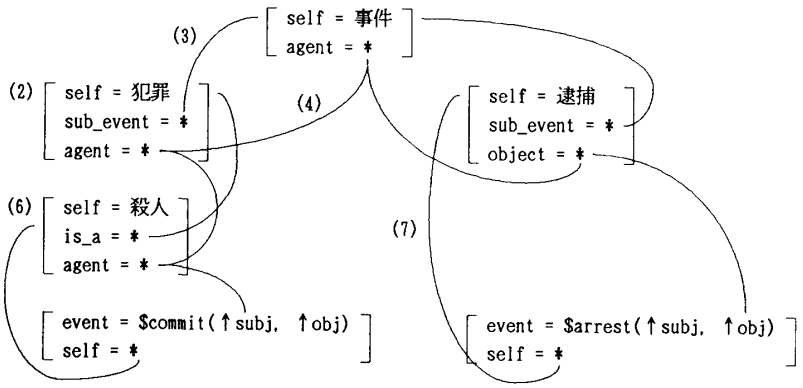

Fig. 7 Functional structure representation of instances.

値が定まった階層的事象構造のことである．このインスタンスは機能構造で表現され，F1, F2 にはそれぞれ「犯罪」，「事件」を表す機能構造が束縛される．

(3) スキーマ F1.sub_event=F2 を評価し，「犯罪」と「事件」の sub_event 関係を表す属性対を機能構造に付け加える．

(4) 次に「犯罪」に付随するスキーマ F1.agent = F2.agent を評価し，「事件」と「犯罪」の行為者が共通であることを表す属性対を機能構造 F2 に追加する．

(5) その後，予測「捜査」，「逮捕」を生成する．

(6) アクター「犯罪」のインスタンスを示す識別子 F1 を受け取ったアクター「殺人」は，(2), (3), (4)と同様にして新たなインスタンスを生成し，辞書アクターに識別子 F3 を送信する．識別子 F3 は，イベントとインスタンスを結合するポインタとして用いられる．

(7) 文(4)の解析で新たなイベントが抽出されると，(5)で生成された予測との間でマッチングが行われる．このマッチングを通じて，階層的事象構造から構文・意味解析過程への文脈情報の伝達が行われる．

以上のようにして，最終的に Fig. 7 に示すインスタンスのネットワークが生成される．また，各インスタンスは，階層的事象構造中のノードと 1 対 1 に対応している．

4. 実　行　例

以下の文章を解析した結果を Fig. 8 に示す．この文章は，Japan Times 1982 年 11 月 30 日号の記事から主要な箇所を抜き出したものである．

(1) Police began a full scale probe into the armed robbery of a supermarket store here late Sunday.

(2) The robbery occurred at around 9:40 p.m. Sunday.

(3) The man threatened Suetoshi with a hand-gun-like weapon.

(4) He demanded the money.

(5) Suetoshi picked up four vinyl bags in the safe which contained the store's weekend proceeds.

(6) He hurled them onto the floor.

(7) The robber put the money in his black bag.

(8) He fled by emergency exit stairs at the back of the building.

(9) Kyoto prefectural police mobilized some 60 officers for investigating the robbery case.

上の文章では，「強盗が起こり，犯人は金を奪った後に逃亡し，警察は捜査を開始した」ということが述べられている．Fig. 8 の階層的事象構造に付随する各番号は，それぞれ文の番号と対応している．文(1)は，イベント「捜査」に対応し，文(2)は，イベント「強盗」に対応している．全体の文章は，「事件」という階層

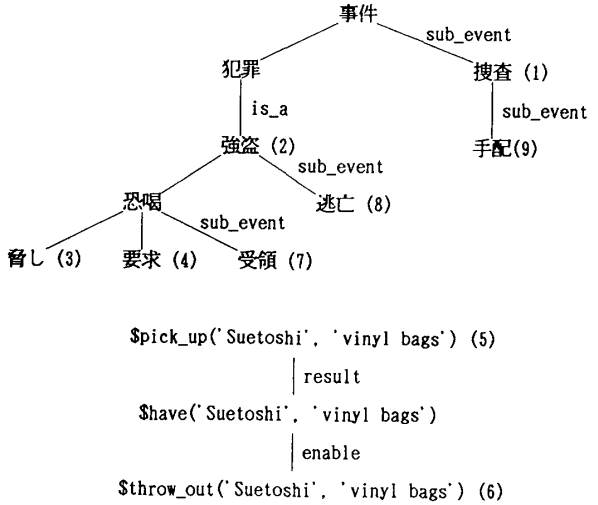

Fig. 8 Contextual structure representation produced by the parse of the sample text.

的事象構造の枠組みでとらえられており，文(3)，(4)，(7)は，「金を奪おうとして恐喝する」一連のイベント系列として，つながりが抽出されている．また，連接関係により文(6)の 'them' が文(5)の 'four vinyl bags' を指していることがわかる．しかしながら，文(5)の関係詞節が，意味的に正しく理解されていないために，'money' が 'four vinyl bags' の中に入っていることを推論できず，文(4)と文(5)のつながりは発見されていない．IP の問題点については 6 節でさらに検討する．

5．階層的事象構造についての検討

IP の予測駆動型解析で用いる階層的事象構造は，スクリプトとフレームの階層性を融合したものとみなすことができる．つまり，Fig. 2 のような枠組みを用意して，場面に固有なイベントの順序関係を推論しようという考え方は，スクリプトの考え方と非常に類似したものである．また，Fig. 3 のように is_a リンクによる階層性を付加し，記憶効率を向上させるという考え方は，フレームの考え方と類似している．

一方，Schank らはスクリプトの考え方を押し進め，人間の記憶構造をモデル化したものとして，MOPs [11] を提案している．MOPs を内部表現として用いた自然言語理解システムに IPP [7] がある．IPP は，入力文を解析する際に，MOPs 中の最も関連の深いシーン（事象を記憶構造内部に組織化するための枠組み）を検索し，その入力を MOPs 内部に組織化しながら文章を理解するものである．さらに IPP では，入力事象間の類似に基づく一般化と，相違に基づく枝分かれの結果として，MOPs 内部の再構成が行われる．

文章理解という観点から眺めた場合には，階層的事象構造は MOPs と同一の役割を持っているということができる．しかしながら，知識表現形式という観点から眺めた場合，両者ともに階層性を表現しているが，MOPs では文脈解析が進むにつれて，構造が動的に再構成されていくのに対し，階層的事象構造では構造を変化することなく，そのインスタンスのみが機能構造として生成されるという点が異なっている．

機械翻訳システム CONTRAST [5] では，MOPs の考え方を工学的見地からさらに発展させた，I-MOP と呼ばれる文脈表現構造を利用している．CONTRAST では，述語の上下関係（一種のシソーラス）を利用して，文の意味表現と I-MOP を照合するという方法で，入力文とシーンを対応づけている．これに対し，IP は，is_a リンクをたどりながら，各イベントの動作主・対象物などに対するスキーマを評価して入力文と照合を行っており，照合方式としては CONTRAST と共通するものがある．

CONTRAST と IP の相違点は，CONTRAST が文の述語を基に照合を行うのに対し，IP では述語を用いて照合できない場合，さらに格要素から得られる情報を基に照合を行っているという点にある．また，文脈解析を起動するための連想キーを，文全体の意味やその文が置かれた文脈に基づいて決定しているために，効率的な探索が行えるようになっている．

6．ま　と　め

本報告では，自然言語理解システム IP の文脈解析メカニズムについて述べた．今後，システム充実のために残されているいくつかの課題について検討する．

〔1〕 4 節の文(5)を構文・意味解析すると，

$pick_up('Suetoshi', 'four vinyl bags') (1)

$contain('four vinyl bags', 'the store's weekend proceeds') (2)

のように，入れ子になったイベントが生成される．事象駆動型解析によって，主節(1)から帰結，

$have('Suetoshi', 'four vinyl bags')

が抽出できるが，前文との連接関係を求めるためには，関係詞節(2)から抽出されるイベントも考慮して，帰結，

$have('Suetoshi', 'the store's weekend proceeds')

を推論しなければならない．しかしながら，現在，IP はこのようなメカニズムを用意していないために，2 文間の連接関係を発見できない．今後は，複数のイベントから論理的に帰結されるイベントも推論できるように，事象駆動型解析を拡張しなければならない．

〔2〕 さらに，文(4)と文(5)から抽出される 2 つのイベント間の連接関係を求めるためには，'money' が 'weekend proceed' と同義であるという，概念間の階層関係が辞書規則中に記述されていなければならない．また，この階層性を扱うことができるように，機能構造間のマッチングを拡張することも必要である．

しかしながら，不用意にこの種の拡張を行うと，マッチングが失敗するたびに辞書中の階層性をたどり，処理効率が極度に低下するという問題がある．文献 (5) では，照応指示の解消にアクティブマッチングと呼ぶ手法を提案しているが，今後このような柔軟なマッチング手法についても考察しなければならない．

〔3〕 本研究では，自然言語理解に必要な基本的意味要素の分類よりも，文脈解析のメカニズムに重点を置いているために，イベントの述語名として単純に基

本英単語を用いるというアプローチをとっている．しかしながら，IP が取り扱う文章の範囲が広くなると，同一事象を表すイベント間で述語名が異なり，マッチングができなくなるという事態が生じる可能性がある．この問題は，Schank らの概念依存理論のように，各単語を概念レベルの基本的意味要素に分解して表現すれば解決できると考えられる．

〔4〕 intersection search の導入によって，時間の流れに関する処理がある程度は可能になっている．たとえば，4節で示した文章のうち，文(1)と文(2)は時間的に逆転しているが，階層的事象構造の交差を発見して，時間的なイベントの流れを抽出している．しかしながら，今後は時間などの概念を積極的に導入して，さらに厳密な文脈解析が行えるようにしなければならない．

〔5〕 プラン・ゴール[10]などを用いた他の推論機能を導入する必要がある．たとえば，'He was hungry. He went to the restaurant.' という文章を考える．第1文の「空腹である」から「行く」というアクションは予測できない．また，第2文の「行く」の前提として「空腹である」というステートを仮定することも不自然である．このような場合，プラン・ゴールを用いた推論を行えば，第1文の「空腹である」から，まずゴール「食事をする」が導出できる．さらに，「食事をする」のプランとして「食堂へ行く」を推論することができ，2文間のつながりを抽出することができる．

◇ 参 考 文 献 ◇

(1) Bresnan, J. (ed.) : The Mental Representation of Grammatical Relations, MIT Press (1982).
(2) DeJong, G. : Prediction and Substantiation ; A new approach to natural language processing, Cognitive Science, Vol. 3, No. 3, pp. 251-273 (1979).
(3) Hewitt, C. : Viewing control structures as patterns of passing messages, Artif. Intell., Vol. 8, No. 3, pp. 323-364 (1977).
(4) Hobbs, J. R. : Coherence and coreference, Cognitive Science, Vol. 3, No. 1, pp. 67-90 (1979).
(5) 石崎 俊，井佐原均：文脈処理技術，情報処理，Vol. 27, No. 8, pp. 897-905 (1986).
(6) Kahn, M. K. : Intermissin-Actors in Prolog, in K. L. Clark and S. -A. Tarnlund (eds.), Logic Programming, pp. 213-228, Academic Press (1982).
(7) Lebowitz, M. : Generalization from natural language text, Cognitive Science, Vol. 7, No. 1, pp. 1-40 (1983).
(8) Lehnert, W. G. : BORIS-An experiment in In-Depth understanding of narratives, Artif. Intell., Vol. 20, No. 1, pp. 15-62 (1983).
(9) Quillian, M. R. : The Teachable Language Comprehender ; A simulation program and theory of language, C. ACM, Vol. 12, pp. 459-476 (1969).
(10) Schank, R. C. and Abelson, R : Scripts, Plans, Goals and Understanding, Lawrence Erlbaum (1977).
(11) Schank, R. C. : Dynamic Memory, Cambridge University Press (1982).
(12) 上原邦昭，垣内隆志，豊田順一：拡張ユニフィケーションを用いたパーザ IP の実現手法，人工知能学会誌，Vol. 1, No. 1, pp. 124-131 (1986).

〔担当編集委員：石崎 俊〕

著 者 紹 介

上原 邦昭（正会員）

昭和53年大阪大学基礎工学部情報工学科卒業．昭和58年同大学院基礎工学研究科博士後期課程退学，同年，大阪大学産業科学研究所勤務．現在，同研究所助手．工学博士．現在，人工知能，特に自然言語理解，および自動プログラム合成の研究に従事している．ACM，電子情報通信学会，計量国語学会，情報処理学会，日本ソフトウェア科学会各会員．

豊田 順一（正会員）

昭和36年大阪大学工学部通信工学科卒業．昭和41年同大学院博士後期課程単位取得退学．同年大阪大学基礎工学部助手，昭和44年助教授，昭和57年大阪大学産業科学研究所教授．工学博士．現在，主として，自然言語理解，画像理解，文書画像処理，およびICAIシステムなどの研究に従事している．電子情報通信学会，日本認知科学会，情報処理学会各会員．

垣内 隆志

昭和60年大阪大学基礎工学部情報工学科卒業．現在，同大学院基礎工学研究科前期課程在学中．自然言語理解に興味を持つ．情報処理学会会員．