考察：「意味理解の一つの手法」

学生時代、言語理論を学んで、すごく面白いと感じました。正規文法とかオートマトン、文脈自由文法、文脈依存文法、プッシュダウンオートマトンとか、なんか数学的に美しい体系になっていて計算機という物を科学だと感じさせてくれました。その後はあまり活用する機会もなく、自然言語処理の仕事をしても、それらを応用しようとすると、現実から遠ざかるような気がして、積極的に利用展開するのはこじつけな気がしてきていました。結局は、お荷物な理論でしかないのかなというのが学生時代から永くの思いでした。でも、この頃、記号処理を考えていくときに基盤となる理論だと悟るに至りました。オブジェクトの入れ子構造は文脈自由文法で捉えるときわめて明確になりますし、プランニングとか、一階述語論理というか、Prolog の仕様なんかも文脈自由文法と捉えると、わかりやすくなります。さらに拡張しようかなんて思ったりもできるようになります。つまり、拡張として文脈依存的に仕様を捉えたらどうなるだろうかという発想にもつながっていきます。文脈依存的な定式化しても綺麗な仕様にまとまるのではないかと。しかも文脈自由型の仕様よりも強力になることは言語理論が明確に語っています。

第１章　意味理解の基本データとは

意味を理解するには次のデータを得ていく事が重要と考えます。

（１）ステージ、シーン

５Ｗ１Ｈの何時、何処でという情報です。時間と場所が大事なのは説明するも無い事ですが、世の中には論理的な場所というものがあることは指摘しておきたいと思います。つまり、会議とかゲームとかもステージとして重要です。野球とか、サッカーとかの活動の枠組みというものは言葉の理解には重要になってきます。「昨日のサッカー、中田選手すごかったな」なんて文は頻発するところです。

理由も論理ステージです。因果の把握は重要です。サッカーの試合のある場面の動きが失点になった、なんてことの把握は生きていく上での重要事項です。因果はある纏まった行動間の関係であるからステージなのです。

（２）アクター

５Ｗ１Ｈの誰がという情報です。問い合わせは誰がやったことなのかという問いが多くを占めます。それにアクターは、カット（活動）の把握には重要な柱となります。

（３）行動

アクターが何したかということですが、行動はアクター間の関係です。「槌で杭をたたく」も、アクターの話者と槌と、杭の間の関係を捉えたものです。この関係として行動を捉えることが重要です。それは視点の変更の要求もありますでしょうし、「あのとき、ああした、あれは」というように、行動を指定してアクターを絞り、指し示すことがあって、それを理解することが必要になるからです。

（４）属性

アクターの属性（形容詞、副詞）は単なる情報以上に、アクターの選別やアクターの意味づけに重要なファクターとなってきます。「どのように」という５Ｗ１Ｈでは How の情報も行動の持つ属性として捉えるとエレガントに扱いができるようになるでしょう。

このように５Ｗ１Ｈを手がかりにして意味理解を考えていくことは見通しを良くするものだと思います。

第２章　Prolog 的な定式化

日本語文章を Prolog 的に定式化することを試みていきます。

基本的には次の形を仮定します。

A:-A11,A12,A13 | A21,A22 | A31,A32.

という様にです。「,」は and 条件で「|」は or 条件です。曖昧性を表現したいので、or 条件は必須になります。コンカレント Prolog にして、or 条件を並列化処理していきたく思います。

A は大抵時間を表す項にすることになると思います。言葉は時系列な表現が基盤だからです。時間を追って、論理が展開していくことになります。

冒頭に「山が動いた」という文がありますと、

Start(*) :-動く（山）,Case(agent,山) ,Tense(past) |動く（%もの）,Attribute(%もの ,not_movable),Case(agent,%もの),Tense(past).

と実際に「山が動く」を表現し、「動かないものが動く」という解釈もあるということも示す Prolog 表現を得ます。

パターンを扱うと、どうしても文脈依存文法な表現が必要になってきます。例えば、「動かないものが動く」は「不可能と思われていたものが可能となった」と理解するわけです。そんな表現は次のように表現するのが素直でしょう。

動く(%もの),Attribute(%もの,not_movable):-CHANGE(%不可能,%可能).

ここで、「%」はシステムが生成した項であることを示します。「*」は変数を表すしきたりだそうですので、推論は「%」にしました。

Prolog のデータベースを扱う命令とかデータエリアのタイプも豊富に必要になってくるでしょう。ステージやアクタープールの生成とかそれらのデータエリアの関係を定義していくこととかやるべきことが沢山あるのです。

その他に推論するために、Prolog の項を書き換えたり、節を追加挿入したりといったことが動的にできなくてはなりません。

第３章　Prolog 的自然言語処理

次の文を考えて見ましょう。

（例文）赤い花が庭に咲いた。

先ずは「赤い」が「花」を修飾し、「咲いた」の格支配が「が」と「に」になることをプレーン文解析で求める必要があります。更に辞書から、「咲いた」のコアとなる格支配は「が：agent」のみであること、「に：location」は一般的な動詞格支配であることを得ます。

それで、次の Prolog 表現を得ます。

:-咲く（花）,Case(agent,花),Tense(past),Attribute(はな,赤い),Case(location,庭),%Stage(location,庭).

そして、背景のデータベースにはオントロジーで、単語「花」、「赤い」、「庭」を定義して持っている事になります。もちろん、「Case」とか「agent」、「Tense」、「past」、「Attribute」、「location」、「Stage」などのキーワードもオントロジーで定義して、システムデータベースに保持していくことになります。

ただ、オントロジーで良いかどうか、この頃迷っています。それは文脈フレームによって意味が異なる単語が沢山あるからです。例えば、「遊ぶ」ですと、演奏とか、短歌会とか、ゲームとか、運動とか、意味が異なります。運動でも、サッカーなのか野球なのかといった問題があります。つまり、「遊ぶ」から連想される状況内容が異なってくるのです。オントロジーでは表現が複雑になるでしょう。

文脈依存文法な Prolog を導入すれば、次のように簡潔に表現できます。しかも、Prolog 節に簡単に組み込み、削除ができます。

Stage(野球),遊ぶ（*person）:-投げる（*person,ボール）,Case(agent,*person),Case(object,ボール)|打つ(*person,ボール),Case(agent,*person),Case(object,ボール)|走る（*person）,Case(agent,*person).

視点の変換も、単語の意味の包含関係の管理も文脈依存文法な Prolog 表現を用いると簡潔な推論を展開できます。Prolog の威力です。

StayRelation(*Object,*REFFER_Object) ,Attribute(StayRelation,右) :-StayRelation(*REFFER_Object,*Object),Attribute(StayRelation,左).

（説明）「REFFER_Object が Object の右にあるとは、Object が REFFER_Object の左にあるということである」と定義しています。

以下、例を上げて答えを探すアイデアを探っていきましょう。

（１）語の欠落対応

（例文１）「瑞樹はテニスをした。水泳もした。それで一日が終わったそうだ。」

この第１文は、次のようになります。

する（瑞樹,テニス）,Case(agent,瑞樹),Case(object,テニス).

この Case(agent,瑞樹)で、Prolog のデータベースにこの主語情報を格納するようにすると良いようです。そのときには、Prolog ライブラリとして、格情報設定節（項）を設ける事になります。

Case(agent,*ND(*:+human)):-put(agent_case,*ND(*:+human)).

第２番目の文の様に主語がない（水泳は ND(*:+human)でない）ので、データベースから読み込むようにします。データベース読み込みも Prolog ライブラリとして用意することになります。

（２）視点変更

（例文２）瑞樹と信子は犬を連れて公園を散歩した。瑞樹が信子の右側を歩くと、犬は左側を歩いた。

（問い）瑞樹の左には誰がいたか？

?:-Stay(*ND(:+person)),Case(with,瑞樹),Attribute(Stay,LEFT),Case(agent,*ND(*+person)).

例文２の第２番目の文は、次のようになると思います。

Scene(DeltaTime):-歩く（瑞樹）,Case(location,Zoon),Attribute(location,RIGHT),Attribute(ZOON,信子).

Cause(current_scene):-歩く(犬),Case(location,Zoon1),Attribute(location,LEFT),Attribute(ZOON1,%信子).

解を求めるために「歩く（*)」を「Stay(*)」に書き換えます。これは Prolog 節・項プールに対して行います。そうすれば、「歩く(*)」が全て「Stay(*)」となります。

そして、Prolog 化文章を実行して、Stay(瑞樹)のものを全て出力して、瑞樹中心の表現に書き換えます。ここでは、

Stay(信子),Case(location,Zoon2),Attribute(location,LEFT),Attribute(ZOON2,瑞樹).

となるでしょう。表現変換はパターン変換辞書を用いてメタな処理として実現します。

結果、「信子」という回答になります。無論、次の規則がメタに辞書にある事が必要ですが。

Case(agent,*ND),Case(with,*ND1),Attribute(Stay,*LEFT_RIGHT):-Case(location,*ZOON),Attribute(location,*LEFT_RIGHT),Attribute(*ZOON,ND1).

（３）問いの対象を求める事

（例文３）「ほとんど怒りに満ちた声を上げた桜井さんの横からファイルを覗くと、コピーを繰り返して文字がかすれ気味で構成もわかりにくい表であった。会社に入ってから今までに見た書類はほぼ全部ワープロで作られていたので､手書きの数字が並んだその紙は､実際の年数よりももっと古いように感じた」

（フルタイムライフ　柴崎友香著　マガジンハウス）

（問い）古いように感じた理由は？

?:-Cause(*why),感じる(%紙),Attribute(%紙、その),Attribute(%紙,古い).

Attrinute(*ND,その):-その（point,*x）,PointFrom(*ND).

その(point,*x),PointFrom(紙):-Search(*x),SameObject(紙,*x).

この文は、「～ので」があるから「会社に入ってから見た書類はほぼ全部ワープロで作られていた」としたら、回答は間違いになります。

「古い」と同じベクトルの属性をもつ、「文字がかすれ気味で構成もわかりにくい」を取り出すと正解になります。問題は「～ので」でなくて、「その紙」が何処を指しているかを決定するアルゴリズムにあるということになります。でも､「その紙」は「ファイル」と「表」と「書類」の３つが候補になり、論理からはどれを選べばよいか分かりません。

結局、回答としては、「紙」の属性を列挙していくことになります。

（回答）a.コピーを繰り返して文字がかすれ気味で構成もわかりにくい。

b.今までに見たものはほぼ全部ワープロで作られていた。

おわり