ICOT Technical Report: TR-508

TR-508

制約と言語

橋田浩一

September, 1989

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

制約と言語

橋田 浩一
新世代コンピュータ技術開発機構 (ICOT) 第2研究室
〒108 東京都港区三田 1-4-28 三田国際ビルヂング 21 階
電話: 03(456)3194, Junet: hasida@icot.junet

概要

AI、特に言語にまつわる情報の部分性を扱うためのプログラミングおよび計算の方法は、(I) 制約の内容は情報の流れから独立にプログラムされ、(II) 計算とはそのようなプログラムの変換であり、(III) 処理の対象は組合せ的な構造を持ち、(IV) 計算はヒューリスティクスに基づいて部分的に行なわれるべきである、という4つの要請を満たす必要がある。これらの要請を満たす、依存伝播という方法を提示する。(I) と (II) により、依存伝播は、一種の制約プログラミングの枠組となる。この制約プログラミングの観点を言語に適用することにより、言語処理過程を様々な制約の統一的な処理として捉える枠組が導かれ、状況意味論などの最近の形式意味論のいくつかの性質が情報の部分性を扱う上で本質的なものとして位置付けられる。また、談話理解システムにおける依存伝播による質問応答の例を示し、そこでデフォルト推論や会話の含意の処理など、情報の部分性に付随すべき情報処理が自然に実現されることを例証する。

1 はじめに

一般に問題は、制約 (constraint)、即ち何らかの対象の構造に関する情報によって規定される。伝統的な意味において、問題を解くとは、その制約を満足するような対象の構造を求めることである。計算機を用いた情報処理技術が成功を収めて来たのは、**完全情報問題**(total-information problems)、即ち、関係する情報を完全に参照できるような問題の領域においてであった。ここで、情報を完全に参照できるとは、第1に情報 (知識と言ってもよい) が完全に与えられており、しかも与えられたその情報を現実的な計算量の範囲内で完全に処理し尽くせるということである。

ところが、世の中の実際的な問題の大部分は、**部分情報問題** (partial-inoformation problems)[1] である。つまり、そこでは、関係する情報を部分的にしか参照することができない。このことを**情報の部分性** (partiality of information)[2] と言う。一般に、部分情報問題は上記のような伝統的

[1] ここで言う完全情報問題と部分情報問題の区別は、Simon [21] の言う well structured problem と ill structured problem の区別にほぼ等しい。同様の分類法として、algorithmic problem と non-algorithmic problem などがある。ここでこうした既存の用語に代わって敢えて「部分情報問題」という言葉を用いるのは、その方が問題の本質をより適確にとらえた言い方であり、後に明らかになるように、AI における困難に対処するための方法を求める上で有益な視点を与えるからである。

[2] 無論、完全情報問題と部分情報問題とは明確に2分されるものではなく、問題全体は、情報の部分性の程度に応じたスペクトラムをなす。本論文で論ずる部分情報問題は、このスペクトラム上で情報の部分性の程度が大きい方の

な意味においては解くことができないから、われわれは、部分情報問題あるいは情報の部分性に「対処する」ないしはそれらを「扱う」という言い方をすることにしよう。情報が部分的であるということは、誤り(偽の信念を持ったり不適切な行動を行なったりすること)を一般には回避できないということであり、情報の部分性に対処するとは、その誤りによる被害を小さくとどめるということである。

情報の部分性には、以下の2つの側面がある(ただし、この2つの側面は実際上は厳密には区別できない)。

知識の部分性 (partiality of knowledge): 関係する情報を完全には知り尽くせない。
処理の部分性 (partiality of processing): 知っている情報を完全には処理し尽くせない。

どちらもごく当り前の事情であり、日常的に普通に生じている。知識の部分性とは、たとえば、明日の天気を知らないということである。一方、処理の部分性とは、極端な例を挙げれば、将棋の規則を知っていても必勝手順がわかるわけではないということである。ロボットにせよ人間にせよその他の生体にせよ、莫大な情報に満ちた世界の中に置かれた有限な行為者にとって、その情報を全て調べ尽くすことができないことは言うまでもなく、ごく部分的に調べればよいにしてもどの部分を調べるべきかが一般にはわからない。このような意味において、AIの問題、即ち、このような行為者の理論が扱うべき問題は、全て部分情報問題である。

ある観点に立った場合、情報の部分性はさまざまな困難の源泉である。その困難とは、曖昧性 (ambiguity, vagueness)、文脈依存性 (context sensitivity)、フレーム問題 (Frame Problem)[3]などであり、これらは、AIの問題に特有の困難の主要な部分を占める。しかし、別の見方も可能である。即ち、表層上は同一の言語表現を多様な文脈において様々な用途に使うことができたり、それに応じて言語が複雑微妙で豊かな意味を持ったりするということも、情報の部分性ゆえであると言える。このような観点は、状況意味論 [1, 2] の立場にも通ずる。

本稿は、AI、特に言語の使用や意味の研究に関して、情報の部分性という観点から得られるパラダイムを提示する。人間の言語使用のメカニズムを明らかにする上での最大の課題のひとつは、情報の部分性から生ずる曖昧性や文脈依存性のような困難を克服することであるが、それは同時に、やはり情報の部分性から生ずる言語の効率性や意味の豊かさを捉えることでもある。以下ではまず、第2節で、情報の部分性を扱うためにはある種の制約プログラミングの方法が必要であることを指摘し、これに沿って、依存伝播というプログラミングおよび計算のパラダイムを提出する。次に第3節で、その議論から得られる言語観を論じ、制約パラダイムと状況意味論などの形式意味論との関係について述べる。第4節では依存伝播による具体的な問題解決の例を紹介し、逆方向の情報の伝達やデフォルト推論など、情報の部分性に対処するための情報処理が依存伝播によって自然に実現されることを示す。

2 プログラミングと制約

一般に、AIの問題領域、特に言語の領域における情報の部分性を計算モデルによって扱うためには、少なくとも以下の4つの要請を満たすプログラミングおよび計算の方法が必要である。

(I) 制約の内容と情報の流れとが互いから独立に記述される。
(II) 計算は制約の変換である。
(III) 一般の組合せ的 (combinatorial) な構造を持つ対象を扱う。
(IV) 計算の制御はヒューリスティクス (heuristics) による。

端に近い、自然言語の処理などの問題である。

[3]フレーム問題と情報の部分性の関係については、松原と橋田 [14] を参照されたい。

以下、これらの条件が要請される理由を各々について論じ、その後で、これらを満たす依存伝播という方法を提示する。

2.1 制約とヒューリスティクス

要請(I)は、制約の内容の記述の中に処理手順の明示的な指定を含めないということである。たとえばPrologの場合には、ホーン節の本体が左から右へと処理されることが決まっているから、この要請が満足されない。制約の内容と処理の手順を混ぜて指定する方法では、部分情報問題を扱うために実現すべき情報の多様な流れを実現できない、というのが、(I)を要請する理由である。ここで情報の流れとは、情報の表現のある部分の情報を参照することによって他の部分の状態が変化することである。たとえば、変数Xの値が2であるという情報を参照し、 Y:=X+1 という代入によって変数Yの値をるとした場合には、XからYに情報が流れたことになる[4]。

問題を規定する制約の内容を記述する際にそれを処理する際の情報の流れをも同時に指定するプログラミング法を、手続き型プログラミングと呼ぼう。(これは一般に考えられている手続き型プログラミングと大体同じものである。)たとえばソーティングの手続きには、求めるべき構造に関する(与えられたいくつかの数値を要素とするリストであって、しかもある大小関係に関してそれらの要素が昇順に並んでいる、という)制約と、その制約を満足する構造を得るための処理手順とが、渾然一体となって埋め込まれている。このようなプログラミングの方法が有効なのは、関連する情報のどの部分が参照可能であるかが決まっており(たとえばソーティングにおいては、普通は順序付けるべき値の集合が所与である)、従って、問題を解く際の情報の流れ方を狭く限定できる場合であり、計算機で扱われてきた問題の多くは、この条件を満たすように作られた完全情報問題である。

しかし、言語の処理を含む多くの部分情報問題においては、情報のどの部分がどの程度参照可能となるかが場合によって異なり、その場合の数は、少くとも関係する制約全体の複雑さの指数関数である。そして、情報の流れは情報が「多い」所から「少ない」所へと向かうから、そのような問題を扱う際の情報の流れの多様性は制約全体の複雑さをはるかに凌ぐ。仮に、「水ちょうだい」という発話の理解を考えてみると、この発話がたとえば台所でなされたという情報が参照可能かどうか、話者が花瓶を持っているという情報が参照可能かどうか、あるいは発話の各部分に関する情報がどの程度参照可能か(たとえば、どの程度はっきり聞き取れたか)など、関連する情報の参照可能性について、潜在的には莫大な数の組合せがあり、それに応じて、この発話を解釈する際の情報処理は、多様な種類の制約のモジュールの間の多様な方向の情報の流れを含むことになる。

この場合に関係する制約のモジュールとしては、語彙、形態論的制約、統語論的制約、意味論的制約、語用論的制約、言語外的な制約があり、さらに、言語外的な制約は、水に関する知識や料理に関する知識などのモジュール群からなる。それ自身がこのように十分複雑な制約の複雑さを遥かに凌ぐ複雑さを持つ情報の流れを明示するプログラムは、実際問題として人間の手に負えないほど複雑なものとなろう。

しかもこの複雑さは、プログラムがモジュール構造を欠くという、たちの悪い複雑さである。プログラムにモジュール構造を持たせ、管理可能にするためには、制約のモジュール構造がプログラムのモジュール構造に反映されなければならない。しかし、手続き型プログラミングにおいてそれを行なうと、情報の流れが制約のモジュール構造に沿うように狭く限定されてしまい、かと言って多様な情報の流れを実現しようとすると、今度はプログラムのモジュール構造が崩れてしまう。たとえば、手続き型プログラミングを用いた文の理解を考えた場合、制約のモジュール

[4]または、情報の表現 X=2 から Y=X+1 へ情報が流れて後者が Y=3 という形になったとも言える。

構造をプログラムのモジュール構造に単純に反映させると、まず構造解析を行ない、その後で意味解析を行なうようなプログラムができる。しかし、たとえば意味的な情報を構文的な情報よりも優先的に用いるべき場合も実際には多いから、このようなプログラミングは情報の流れを不当に狭く限定していることになる。そこで、たとえば構文解析の途中で意味的な情報の一部を参照するというようなプログラムを作ることになるが、これはプログラムのモジュール構造を崩すことだから、この路線を推し進めて行くと、システムはじきに複雑過ぎて管理不能なものとなる。こうして、手続き型プログラミングによって本格的なAIシステムを開発しようという企ては全て破綻する。機械翻訳システムやエキスパートシステムなど、大きなAIシステムにおいて、処理の範囲を少し広げようとするとたちまちプログラムが複雑化し、拡張はおろか保守することすらできない代物になってしまう、という話をしばしば耳にするが、それはこのような事情による。

この複雑化の原因は、制約の内容と情報の流れを混ぜて記述したことにある。従って、多くの部分情報問題の処理において、プログラムのモジュール構造を保証しつつ多様な情報の流れを実現するには、制約の内容と情報の流れとが互いから独立に記述されなければならない。以上をまとめると、どのように情報を流せばよいかを予め限定できる場合には手続き型プログラミングによって制約の各部の内容に対する情報の流れを時前に指定しておくことができて、その方が実行の効率がよいが、言語処理のようように情報の流れを時前に限定できない場合には、制約の記述には情報の流れの指定が含まれないようにしておき、実際に処理するときに情報の流れを動的に決定せざるを得ない。

こうして要請(I)が導かれたが、そこから直ちに要請(II)が得られる。即ち、(I)により、いわゆる知識や信念などの、世界に関する情報が常に制約の形で記憶されるから、システムに内在化された情報の処理とは、新しい局面により効率的に適応できるような形に制約を変換することである。情報の部分性により、対象の構造を一般には一意に決定できない、ということも考えると、部分情報問題において処理されるべきものは制約であって対象ではない。情報処理はその次の場面においてより適切に行動するためになされるものだから、そこで第一義的に重要なのは、対象の構造ではなく、(思考を含む)行動に課せられるべき制約である。対象の構造は制約の一部と見なせる、と言う意味においても、構造は副次的なものに過ぎない。むしろ、制約のうちで構造を表現する部分とそうでない部分の間に明確な区別を設ける必要がそもそもないと考えるべきであろう。

また、要請(I)から、(II)で言う処理過程は多様な情報の流れを実現しなければらない、ということも直ちに導かれる。つまり、(II)の処理過程が実現する情報の流れがある方向に偏ったものであれば、制約の記述の中に情報の流れが含まれることになってしまう。たとえば、Prologの処理過程を考えると、ホーン節の本体が常に右から左に処理される等の事情によって情報の流れに偏りが生ずるから、Prologのプログラム中には制約各部の内容とともにそれを処理する際の情報の流れも指定されてしまう。

要請(III)は、対象の担う情報への参照の容易さに関する要請である。まず、情報が部分的にしか得られない場合には、部分的な情報を容易に参照できることが望ましい。即ち、制約の構造は、部分を単に繋ぎ合せることによって全体が構成される、という意味において**組合せ的**(combinatorial)[5]であるべきである。組合せ的構造とは、最も一般的にはグラフであり、意味ネットワークやフレームなども組合せ的である。このような構造は、各部分に自明な仕方で言及することができるという意味で、部分情報問題の処理に適している。たとえば、$p(X) \wedge q(X)$のような論理式によって

[5]組合せ的という言葉を用いるのは、このような構造がいわゆる組合せ問題(combinatorial problem)と深い関係を持つからである。これに対し、合成的(compositional)という言い方を用いなかったのは、関数の合成(composition)などとの連想を避けるためである。

表現された制約は、p(X) および q(X) という部分を自明な方法で繋ぎ合せることによって構成されているという意味において組合せ的であり、これらの部分を簡単に参照できる。ところで、対象の構造もまた上述のように制約の一部と考えられるから、同様に組合せ的となる。即ち、ここで考えるべき対象は、一般にネットワークや木のようなものであって、たとえば実数や有理数などに限定すべきではない[6]言語は組合せ的な構造を持つが、人間の知能がそのような構造を用いているのも、情報の部分性に対処するためであろう。

要請 (IV) に言うヒューリスティクスは、情報の欠如を補うためのものである。処理の部分性として指摘したように、与えられた制約は完全には処理し切れないから部分的に処理する必要があるが、どの部分をどのような順序で処理すればよいかを決定するための情報は一般には得られない。従って、部分的処理の制御は、必ず正しいとは限らない判断を含む。即ち、処理の優先度に関するヒューリスティクスを用いることになる。さらに、そのようなヒューリスティクスがあれば、部分的処理を制御するのみならず、(暫定的な) 結論を導くためにも用いることができる。即ち、たとえばイェール射撃問題 (Yale Shooting Problem) [7] のように、2 つの競合する結論のうちいずれを選ぶべきかが完全にはわからないけれどもとりあえずどちらかに決めておく、という際に用い得る。知識の各部分にたとえば確信度 (certainty factor) のようなものが割り当てられているとすると、知識の中のいくつかの部分が連言 (AND) で繋がっているときには確信度の小さな部分から優先的に処理し、選言 (OR) で繋がっているときには確信度の大きな部分から優先的に処理すればよい。競合するいくつかの結論から暫定的に 1 つを選ぶ際にも、確信度の大きなものを選べばよい。(後に導入するヒューリスティクスは確信度のようなものも用いているが、それだけではない。)

ヒューリスティクスは、制約の内容とは別に与えられるのではなく、制約から導かれるものである。制約は論理によって記号的に分節される側面、つまり組合せ的な構造であるネットワークと、そうでない側面 (後に述べる定式化では、このネットワーク構造上のエネルギーの分布) とからなり、ヒューリスティクスはこれらの組合せから生ずる。また、制約にこのような記号的でない側面を考えざるを得ないのは、制約が完全に記号的だとすると、それは一般には完全に処理し尽くせず、従って、部分的に処理するためのヒューリスティクスが必要になるからである。記号的な制約を処理するためのヒューリスティックな規則をやはり完全に記号的に分節できるような形で与えようとすると、その規則に関しても部分的な処理の必要が生じるから、さらにそのためのヒューリスティクスが要ることになり、無限後退に陥る。これは、そもそもシステムの仕様記述が完結しないという問題であり、処理が完結しないという問題とは異なることに注意されたい。

これに関連して注意すべきは、ヒューリスティクスは制約の一部なのだから、その多くの部分が問題領域に依存する、ということである。従って、ヒューリスティクスを領域知識の一部としてプログラムすることができなければならない。たとえば Simon らの一般問題解決器 (GPS; General Problem Solver) は、手段 - 目的解析 (means-ends analysis) のような、領域に依存しない一般的なヒューリスティクスを用いていたが、これでは不十分であり、GPS は非常に限定された簡単な問題にしか適用することができなかった。この失敗を繰り返さないためには、領域知識の一部としてのヒューリスティクスを問題解決過程に反映させられる必要がある。

以上の要請、特に (I) と (II) とは、**制約プログラミング** (constraint programming) に通ずるものである。しかし、これまで行なわれて来た制約プログラミングの研究[4, 6, 19, 23] は、要請 (III)、(IV) を満足しようとするものではない。(III) に関して言うと、他の制約プログラミングのアプローチにおいて要請 (I) と (II) を満たすような形で扱っている対象は、数値やブール値、あるいは有限領域内の対象である。組合せ的対象の領域である Herbrand 領域などに関する制約

[6]無論、数値的な対象も組合せ的な構造として表現することにより対象の一種とすることが可能である。

についての研究も、最近の制約プログラミング以前にあった [17, 18] が、そこでは、情報の流れをかなり明示的に指定する、手続きパラダイム寄りの方法が用いられており、(I) と (II) が十分に満たされていない[7]。また、制約プログラミングに関する従来の研究は、ヒューリスティクスを本格的に論じてはいなかった。これは、組合せ的な対象を扱っていないことにも関係している。即ち、組合せ的な対象に関する制約充足問題の多くは原理的に計算不可能なものであるから、そうした問題を実際的に扱うにはヒューリスティクスが不可欠となる。だからこそ、制約プログラミングの研究のほとんどは組合せ的な領域に関する制約の扱いを避けているとも言える。

2.2 依存伝播

ここで、上の 4 つの要請に従って、部分情報問題を扱うためのプログラミング・パラダイムを導入する。後に詳述するように、このパラダイムにおける計算は、ネットワークのある部分に生じた依存関係なるものが処理を引き起こし、それによってその周りに別の依存関係が生じ、それがさらなる処理を引き起こす、という形で進行する。そこで、これを**依存伝播** (dependency propagation) [10, 12] という[8]。要請 (I)、(II) を満たすことにより、依存伝播は制約プログラミングの一種となる。

依存伝播の処理方式を具体的に論ずる前に、その背景にある直観を述べておこう。依存伝播は、動物 (特に人間) の知能における記号処理 (symbol processing) のモデルである。まずは常識通り、神経回路の興奮パターンのマクロな構造として、脳の中に記号[9]のネットワークがあると考える。神経回路の興奮パターンの変化に伴って、このネットワークの構造も変化する。これが脳における記号計算である。状態変化を起こす系の一般的な性質として、ある種の**ポテンシャル・エネルギー**がこのネットワークの各部において定義され、そのポテンシャル・エネルギーが高いほどネットワークの各部の局所的な安定性が低い、というふうになっているはずである。即ち、記号のネットワークの各部分は、そこでのポテンシャルが局所的に減ずる方向にその構造を変えてゆくことになる。

この記号のネットワークが何らかの制約を表現していると考えることにより、先に論じた 4 つの要請のうち、(I) が満たされ、これによって (II) も満たされる。計算機上のモデルとして実現する際には、その制約を 1 階述語論理の式によって表現することにする。さらに、その制約を課せられる対象を Herbrand 空間の対象 (ただし巡回的 (cyclic) な構造も許す) と考えれば、要請 (III) も満足される。さらに、ネットワークの構造変換は、一般的な導出原理 (resolution principle) による推論を含むような処理の並列実行に相当するような、プログラム変換ないしは部分計算であると考える。要請 (IV) のヒューリスティクスは、**依存関係** (dependency) とポテンシャル・エネルギーによって実現される。即ち、制約のネットワーク中でポテンシャルが高い部分が優先的に処理される。

たとえば Prolog のような順序付きの入力導出に限定せず、一般的な導出法を含むような推論の方法を採用したのは、多様な情報の流れを実現するためである。情報の流れが多様であるということは、任意の文脈において情報処理が無制限に行なわれるということではなく、適当な文脈を与えれることによって生じ得る情報の流れが、全体として多様である、ということである。つまり、各々の文脈においては、処理の範囲は厳しく制限されなければならない。そのために、制

[7] この議論に関しては、[12] を参照されたい。

[8] 依存伝播を伴う単一化を、条件付単一化 (conditioned unification) [11] または制約単一化 (constraint unification) [26] と言う。

[9] ここで言う記号とは、それ自身の内部に意味を持つものではなく、内部構造を持たない単なる論理変数のようなものである。こうした記号の「意味」は、その記号がネットワーク中のどこにあり、どのような形で情報処理に関わるか、さらにはシステム全体が外界とどのように関わるかに依存して生ずる、随伴的な属性である [16]。

[illegible]を制御するためのヒューリスティクスが必要となる。一般[illegible]と言われているが、それは適切な制御構造を欠いていたから[illegible]た論理を用いるのは、それが今の所、組合せ的構造に関する[illegible]であるからに過ぎない。実際には、処理は部分的に[illegible]完全性も無矛盾性も保証されない。ゆえに、プログラ[illegible]論によって正確に与えられる、ということはない。

[illegible]ログラム(制約)はプログラム節の集合として表現される。また、Prolog を[illegible]必要はないし、また、閉世界仮説は必ずしも用いられない。[illegible]否定、束縛、および束縛の否定をひっくるめて要素制約 (atomic constraint) と呼ぶ。[illegible]制約と対象の構造を特に区別しないということである。また、変[illegible]と呼ぶ。節は有限個の要素制約の連言である。プログラム中に、先頭節 (top clause) と呼ばれる特別な節がただ一つあり、これは Prolog の問合せ (query) に対応する。[illegible]は、Prolog と同様、先頭節の否定 (存在限量化された要素制約の連言) を証明することである。即ち、先頭節の否定を証明するか、または先頭節の否定が正しいときの abductive な説明[illegible]を与えることである。そのために、節を用いてある種の変換を行ない、その変換によって得られる要素制約の連言を証明しようとする、ということに関して [illegible] Prolog と同じである。

依存関係 (dependency) とは、証明すべき要素制約の連言に現われる2つの(同一かも知れない)要素制約の間の[illegible]依存関係のことである。たとえば、p(X,X)、p(X,Y) ∧ q(Y)、お[illegible]関係がある。依存関係によって、その依存関係を解消するための処理が[illegible]。その際、[illegible]その部分の周辺にしばしば新たな依存関係が生じ、これ[illegible]する。そのときの各処理は、展開 / 畳込み (unfold/fold) [illegible]の因子化 (factoring) である。依存関係[illegible]ヒューリスティックが、制約の種類から独立に抽象的に規定[illegible]。

[illegible]例をいくつか挙げる。(1) のような先頭節は述語[illegible]、このとき、新述語 r は (4) のように定義される。

[illegible]

[illegible]ことに注意されたい。もう少し詳しく見ると、まず[illegible]に基づいて下のように定義される。

[illegible]
p(A) :- memb(A, [b])

ここで[illegible]ているから、同様にこれを解消すると、この節は (6) のように[illegible] q を (7) のように定義される。

(6) p(A) :- q(A).
(7) q(b).

q の定義は[illegible]、(6) において q(A) を A=b で置き換えると、q の定義は (8) のようになる。

別の例を挙げると、(8) は (9) に変換され、述語 r は (10) のように定義される。また、(11) は (12) に変換される。

```
 (8) :- memb(X, [a, b, c]), memb(X, [b, c, d]).
 (9) :- r(X).
(10) r(b).
     r(c).
(11) :- memb(X, [a, b]), memb(X, [b, c]).
(12) :- X=b.
```

以上の例からわかるように、依存伝播によって対象の構造に関する可能性が絞られて行くことになる。

これらの例のように、ホーン節論理で閉世界仮説を用いた場合、変数に関する依存関係があるということは、制約のその部分が充足不能かも知れないということであり、逆に、全ての述語が本体にそのような依存関係のない定義節を持つならば、先頭節の否定は充足可能である (最大モデルを持つ)。従って、閉世界仮説付きのホーン節論理の場合、変数に関する依存関係を解消するというのは、非常によい処理のヒューリスティックである。依存伝播が処理する制約はホーン節論理に限定されず、しかも閉世界仮説を用いるわけでもないが、その場合でもこれは依然としてかなりよいヒューリスティックと考えられる。また、述語と関数記号の共有をも依存関係と見なすのは、たとえば :- p, p. のような冗長性や :- X=a, X≠a. のような矛盾を処理するためである。

制約の各部分にポテンシャル・エネルギーというアナログ量が分布しているとする。そして、依存関係のみならずポテンシャル・エネルギーをも用いるような、よりきめの細かい実行制御のヒューリスティックを考える。ポテンシャル・エネルギーは、**活性度** (activation) と**仮説コスト** (assumption cost) [13] という 2 つの成分の和であり、制約の各部分に割り当てられる。活性度は注意 (attention) の度合ないしは重要度を表わし、制約のあらゆる部分に割り当てられる。一方、仮説コストは要素制約に割り当てられ、その要素制約の成立の疑わしさを表現する。

制約は要素制約、対象、節などからなるネットワークと見なせるが、このネットワーク上で活性拡散 (spreading activation) が生じ、ネットワーク中のリンクを伝わって上記の活性度が伝播する。この拡散のもとになるのは、依存関係から生ずる活性度である。変数に関する依存関係は、2 つの引数位置がその引数の束縛に対して課する制限の強さに応じた活性度を生む。上の (1) から (3) のような変換を行なうことによって依存関係が解消されると、その分だけ活性度が下がる。一方、述語および関数記号に関する依存関係は、2 つの要素制約の仮説コストの差が大きいほど大きな活性度を生ずる。これら 2 つの要素制約を因子化すると、その結果できる要素制約の仮説コストは、もとの要素制約の仮説コストの低い方に一致し、こうして全体として仮設コストが下がる[10]。また、同時に依存関係が解消されるので、活性度も下がることになる。(依存関係を解消するような処理の他に、活性度の高いリテラルを実行または展開して活性度を下げるという処理もあるが、その詳細は後に示す例題に譲る。)

処理は一般に制約の構造の中で活性度の高い部分に関して優先的に行なわれる。活性度は重要度を表わすと上で述べたが、それは、活性度の高い部分を処理することが重要だという意味である。活性拡散によって、制約の中で活性が高い部分に近い部分や、他の多くの部分と繋がっている部分の活性が高くなるが、これは、重要なものに関係しているものや、多くのものに関係しているものは重要だという直観に対応する [9]。さらに、活性度は上記のように変数に関する依存関係や仮説コストによって決定されるから、重要度の評価には制約の充足不能性も反映される

[10] Hobbs [13] も同様の方法を用いている。

ことになる。制約の充足不能性の高い部分を処理することが重要なのは、それによって制約の中に含まれている選択肢のいくつかが除去され、より特定的な制約の表現が得られる可能性があるからである。情報の部分性に対処する上での活性拡散の利点は、ネットワークの構造に沿った多様な方向の情報の流れを簡単に実現できる点にある。この理由により、活性拡散 [27, 28] やその記号的近似であるマーカ伝達 (marker passing)[3] は、知識のネットワーク中での検索を行なうために用いられているが、従来は活性度に充足不能性が反映されていなかった。逆に、たとえばHobbs [13] のシステムでは、仮説コストによって充足不能性を考慮しているが、活性拡散を行なっていないため、情報の流れを限定することになっている。

3 制約としての言語

ここでは、以上で述べた枠組を主として言語に対して適用する。まず、言語処理過程を依存伝播の考え方によって捉えることにより、今まで工学的に実現されていたモデルよりも言語現象の実際に近く、しかもソフトウェア工学的にも見通しのよいモデルを提出する。さらに、理論言語学や形式意味論などに関しても、ここまでの議論が有意義な観点を与えることを示す。

たとえば文の理解においては、まず表層文の形態素解析を行ない、次にその結果に基づいて構文解析を行ない、それによって得られた構造を用いて意味解析を行なう、という考え方は、今となってはかなり古臭い言語処理過程の捉え方である。しかしこの考え方は、形態論的制約、統語論的制約、意味論的制約などからなる制約のモジュール構造を手続き型プログラムのモジュール構造に自然に反映させた結果である。このため、工学的に実現された自然言語処理システムの全てが実際には今なおこの考え方に基づいている。

ところが、情報の流れはこのような方法によって実現されるよりもはるかに多様である。たとえば文の理解においても、統語的な情報が不十分であるために統語的な情報のみに基づく解析から莫大な曖昧性が生じ、全ての可能性を記憶しておけない場合には、意味的な情報を優先的に用いざるを得ない。また、たとえば「氏と育ちはあざなえる縄である」という文を読んだときに「縄」を「蝿」と読んでしまう、などという間違いは、人間の言語処理過程が多様な情報の流れを伴うことの証拠である。即ち、この読み間違いは、「うじ」という音韻的情報、蛆が育つと蝿になるという言語外的情報、「縄」という文字と「蝿」という文字がよく似た字形を持っているという視覚的情報などが複合して生ずるものと考えられる。

上記のような手続きのモジュール構造は、実際上、制約のモジュールの間でのこのような情報の多様な流れを実現できない。無理に実現しようとすると、たとえば、構文解析の途中に意味的な情報の処理を挿入するというようなプログラミングを行なうことになる。しかし、前に指摘した通り、この路線を押し進めることによって構文解析と意味解析とを本格的に融合しようとすると、たちまちプログラムのモジュール構造が崩れ、人間の手に負えなくなってしまう。従って、現実の言語処理システムでは、意味的処理のうち予め定められた一部分だけを構文解析の中の定められた所に挿入するという形に、構文的処理と意味的処理の融合を限定せざるを得ない。これは、扱い得る文脈の範囲を不当に狭く限定することになる。制約の種類ごとに手続きを割り当てるという手続きプログラミングは言語処理においても不適切なのである[11]。

この議論は、計算機のプログラミングに限った話ではなく、人間の認知システムにも全く同様にあてはまるはずである。即ち、知識を用いる際の情報の流れの方向が、知識の表象 (representation) の各部ごとにその内容に依存した形で指定されているとすれば、表象全体の複雑さは知識の宣言的な内容の複雑さを遥かに凌ぎ、脳の容量すら大きく上回るものとなろう。無論、処理手順と分かち難く結び付いたいわゆる「手続き的」な知識もあり得るだろうが、全ての知識、

[11] たとえば、名詞句の指示対象を同定するための手続き [15] などを個別に考えるのは誤った方法である。

特に言語のように極めて文脈依存性の強い領域に関する知識までをも、処理手順込みで表象するのは到底不可能である。即ち、脳内の「プログラム」においても、知識や信念は多くの場合、処理手順とは独立に記述されており、これによって前記のような多様な方向の情報の流れがもたらされているに違いない。そして、人間の言語処理過程がそのような処理手順から独立な「プログラム」を用いて行なわれているとすれば、その「プログラム」は当然、文の理解と表出の両方において共通に使われていなければならない。実際、文の理解と表出とが共通のプログラムによって行なわれていると仮定することにより、様々な言語現象に関する認知科学的な説明が得られる[12]。

要請 (II) をそのまま言語に引き写すと、言語処理過程は制約の変換だということになる。つまり、たとえば文の理解とは、先行する談話によって与えられる制約と文の表層の形に関する制約とを合わせたものに変換を施して新しい制約を得る過程である。構文木などの統語構造、あるいは意味構造ですら、そこに明示的に現われる必要はない。文を解析する目的は、意味構造を求めることでもないし、ましてや統語構造を求めることではない。2節で指摘したように、重要なのは対象の構造ではなく、行動に課せられるべき制約であり、これは文の理解においても表出においても同様である。

このことは、曖昧性の処理に関する知見を与える。まず第1に、対象の構造とは何かということが明確に定義されているとしても、曖昧性を解消して構造を特定する必要はない。さして重要でない曖昧性は、解消せずにそのまま制約として残しておけばよい。たとえば、「眠っていた犬を殴った」という文の解釈においては、「眠っていた」時は発話時に対して相対的にそれ以前なのか、それとも「殴った」時に対して相対的なのか、という意味的な曖昧性が生ずるが、どちらの解釈をとるべきかは通常の文脈ではどうでもよい場合が多いだろう。このように、人間が気にしないような曖昧性は単に解消されずに残ると考えればよい。これは、計算とは制約の変換である、という要請の1つの含意である。また第2に、もしも構造とは何かということが明確に定義されないとすれば、そもそも構造を決定するということが意味を失い、従って曖昧性の解消とは何かということ自体が文脈に依存する。曖昧性の解消とはいわば玉葱の皮剝きであり、どこまで剝いてもきりがないからどこかで剝くのをやめなければならないのだが、どこでやめればいいかは場合によって変わる。特に意味論や語用論の領域では、ある部分制約の示唆する可能性を次々に際限なくたぐって行くことができるので、どこかで思考停止しなければならない。どこで思考停止すればよいかという問いに対しては、それは文脈に依存する、としか答えようがない。このようなことは、統語論や形態論の領域でも言えるかも知れない。

また、(II) のような考え方に従えば、全ての発話は自然に発話行為 (speech act) として処理される。即ち、発話の機能が (思考を含む) 行動に対して制約を課することとして捉えられ、その場合の発話の「意味」は、その発話によって課される新たな制約と考えられる。たとえば疑問文の「意味」は、その答えに相当する発話を聞き手が行なうべきである、という制約であり、平叙文の「意味」は、その命題内容に相当する信念を聞き手が持つべきである、という制約である。発話の命題内容、発語内的な力 (illocutionary force)、発語外的な力 (perlocutionary force) などは、それぞれ制約の一部として位置付けることができる。また、制約を課することが発話の機能だという考え方は、状況意味論で言う意味の関係理論 (relation theory of meaning) [1] とも通底する。意味の関係理論とは、意味とは状況 (正確には状況の型) の間の関係だという考え方であり、その関係は、状況理論 (状況意味論の基礎にある数学理論) における制約として定式化される。

意味とは制約である、という考え方は、意味を対象として捉える外延的な意味論よりも一般

[12]詳しくは [8] を参照されたい。また、計算モデルとしての実現に関する議論については、[10]、[20] などを参照のこと。

的な意味論を導く可能性がある。たとえば、制約は曖昧性を包含するから、曖昧性まで含めた上での意味(極端な例としては、掛け言葉の意味)を考えることができる。しかし、意味を制約と見なす考え方そのものは、それだけでは外延的な意味論に帰着させ得るものである。より一般的な意味論の可能性は、制約としての意味という静的な視点を越えて、情報処理システムの動的な側面、即ち、情報処理の部分性の内に見出される。この動的な意味論においては、対象としても制約としても、意味を静的に捉えるのは不可能である。なぜなら、前述のように、そのようなシステムは論理的な矛盾を含み得るものであり、一般には外延的なモデルを与えることができないからである。そこでは、情報の流れ全体を意味と考えるのが最も適切であろう。

要請(III)に関して言えば、状況意味論、談話表示理論、メンタル・スペース理論などの、最近の形式意味論で用いられている意味論的対象は全て、本稿における意味において組合せ的である。特に、状況意味論(正確には状況理論)における事実や事態の構造は論理学の項や式のような組合せ的なものであり、談話表示理論における談話表示構造や、メンタル・スペース理論におけるメンタルスペースが作る構造は、まさにネットワークである。

以上見たように、状況意味論などの最近の形式意味論は、言語における情報の部分性の扱いにおいて本質的な様々の性質を持っている。しかし、情報の部分性の主要な原因の一方は処理の部分性であり、さらに、知識の部分性と処理の部分性は実質的に区別不能である[14]から、情報の部分性を正しく捉えるには、要請(IV)を満たす依存伝播のような計算理論が不可欠である。意味論や語用論のみならず、形態論や統語論においても、そのような計算理論が必要とされる可能性は大きい。

4 例題

ここでは、依存伝播を用いて状況理論および状況意味論を計算機のプログラム中に実現した例を紹介する。状況理論の対象や制約を1階論理で表現することになるが、これは、状況理論を1階論理に帰着させるという意味ではない。以前に注意したように、ここではTarski流のモデル論的意味論に正確には従わない。完全な処理を行なうわけではないから、制約のシステムが無矛盾であることは保証されず、従って、それに外延的なモデルを与えることは一般にはできないのである。ただし、この性質は状況理論そのものによって説明されるものではなく、依存伝播のような動的な計算理論によって初めて実現可能となる。

ICOTでは、DUALSという談話理解実験システムを開発中であり、以下で示すのは、その第3版であるDUALS-III [22]による質問応答の例である。DUALS-IIIは、小学校6年生の国語の教科書の文章を読んで「理解」し、その内容に関する質問に答えるシステムであり、その問題解決モジュールにおいて依存伝播を用いている。ただし、残念ながらDUALS-III全体は、構文解析や文生成には個別の手続きモジュールを用いるという手続き型のパラダイムに基づいている。

DUALS-IIIの処理対象となるある文章は、「人間が、この地球の上で生き続けていくためには、どうしても、自然の恵みに頼らなければならない。」という文で始まる。この文章を読んだ後、DUALS-IIIに「人間はどこで生きていますか。」という質問をしてみる。「地球に生きている」などというのが期待される答であろうが、上記の第1文から得られる情報からこの答を論理的に演繹することはできない。にもかかわらず、DUALS-IIIは「地球に生きている」と答えることができる。いかにしてこの答えに至るかを以下で述べよう。

DUALS-IIIは初め、下のような要素制約を持っている。

(13) `Sit=ground`
(14) `hold(Sit,earth)`

(15) listen(Objs0)

これらの要素制約は、全て先頭節(の否定の本体)に現われる。以下、特に明示しない限り、要素制約は先頭節にあるものとする。Sit は現実に対応する状況(situation)であり、この状況に、earth(「地球」の指示対象)などの個物(individual)が存在し、事実(facts)が成立する。Sit が ground に束縛してあるのは、他の対象と無闇に単一化させないためである。対象 ϕ が状況 ζ に存在することも、事実 ϕ が状況 ζ で成立することも、共に要素制約 hold(ζ,ϕ) で表現する(このように「もの」と「こと」を統一的に扱っている点は元祖状況理論と異なる)。このように、状況の内部構造はその状況に対する制約として表現される。また、この段階において Sit で成立している事実は、DUALS-III の持つ事前の知識を構成するものであるが、ここでは省略した。

DUALS-III では、会話の進行も制約の変換と見なされる。要素制約(15)は、DUALS-III が、入力文を受け取ることによって外界の情報を取り込もうとしている、ということを表現している。ここで Objs0 は、その入力文中に含まれる照応表現の指示対象となり得る対象や事実を要素とする集合である。ただしこの Objs0 は、会話の開始時のものなので空とする。一般に、listen(ξ) の形の要素制約を処理することによって、入力文 s が読み込まれて解析されると、この要素制約はその文の内容に相当する制約やその他の制約に変換される。その結果の制約の中には、listen(η) または speak(ψ,η) の形の要素制約が含まれており、η は ξ の要素(の一部)の他に、文 s によって導入された対象や事実を要素とする。ここで、speak(ψ,η) は、DUALS-III が ψ という内容にあたる発話を行なう、という制約である。

まず DUALS-III は、入力された上記の第1文を読み込んで解析する。この処理は、上の要素制約 listen(Objs0) のポテンシャルが、他の要素制約のそれに比べて十分高いことによって生ずる。述語 listen は手続きとして定義されており、これを処理するということは、文解析モジュールを呼び出すということに相当する。この処理によって、listen(Objs0) が次のような要素制約に変換される。(他にも要素制約が生成されるが、全部は示さない。以下同様に省略する。)

(16) hold(Sit,Purpose)
(17) hold(Sit1,Continue)
(18) presuppose(Sit1,Sit3)
(19) hold(Sit3,Loc1)
(20) hold(Sit3,Live)
(21) Purpose=infon(purpose,[Sit1,Sit2],yes)
(22) Continue=infon(continue,[Sit3],yes)
(23) Loc1=infon(loc,[earth,Live],yes)
(24) Live=infon(live,[human],yes)
(25) Sit1=sit(Sit,Purpose,1)
(26) Sit2=sit(Sit,Purpose,2)
(27) Sit3=sit(Sit1,Continue,1)
(28) listen(Objs1)

Sit1、Sit2 および Sit3 は現実の状況よりも下位の状況である。Purpose、Live などは事実であり、何らかの状況において成立(hold)している。これらの要素制約の間の関係を図1に示す。

このとき、DUALS-III に対して「人間はどこで生きていますか。」という質問をすると、同様に(28)が処理され、下のような要素制約に変換される。

(29) hold(Sit,Loc2)
(30) hold(Sit,Live)
(31) Loc2=infon(loc,[Where,Live],yes)

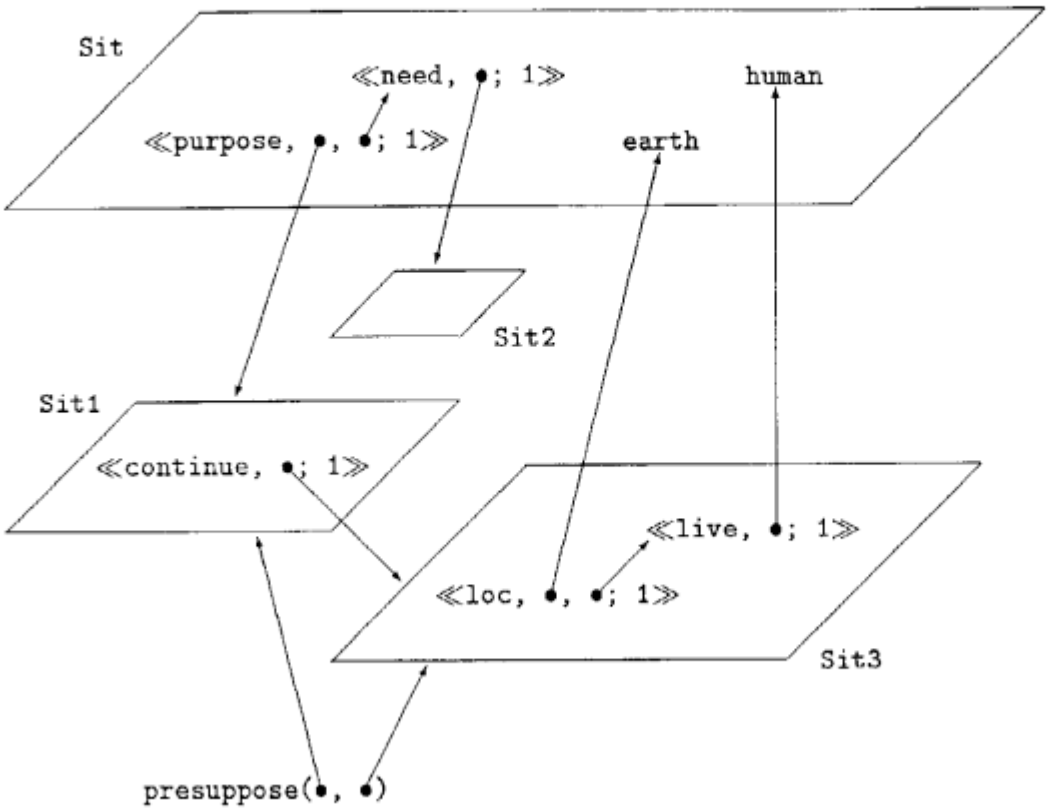

図 1: 「人間がこの地球の上で生き続けてゆくためには、 ··· 」に対応する意味構造

述語 presuppose は、下のように定義される。

(32)
```
presuppose(S1,S2) :- sub_sit(S2,S),project(S1,S).
project(sit(S1,Infon,ArgPlace),S) :- project(S1,S).
project(S,S).
```

ここで sub_sit(S2,S) は、S2 が S の部分状況であること、即ち、hold(S2,σ) なる全ての σ に関して、σ のコピー τ が hold(S,τ) を満たすことを意味する。ただし、σ が構造を持つ場合、その構造中に hold(S2,α) なる α が現われるならば、τ の構造の中の対応する位置には α のコピーが現われるものとする。(18) は、Sit3 が Sit1 または Sit の部分状況であるという意味である。この要素制約は、動詞「続ける」に付随する**前提** (presupposition) であり、下に示す「続ける」の語彙項目から得られる。

(33)
```
lexicon("続け",Category) :-
                ⋮
        index(Category,Fact),
        situation(Category,Sit),
        subcat(Category,[Complement]),
        situation(Complement,Sit0),
        Fact=infon(continue,[Sit0],Pol),
        hold(Sit,Fact),
```

```
presuppose(Sit,Sit0).
```

この語彙項目は、実際には文解析モジュールが用いている辞書に書かれており、表層的には上記とかなり異なるが、内容はほぼ同じである。

一般に、平叙文の意味構造に関する制約は、仮説コストが低く、質問文の意味構造に関する制約は仮説コストが高い。たとえば、(16)~(27) の仮説コストは低く、(29)~(31) の仮説コストは高い。前述のように、仮説コストの高い要素制約を仮説コストの低い要素制約と因子化することによって、全体としてポテンシャルを下げることができる。

この例の場合には、(23) と (31) を因子化することができる。しかし、(29) と (30) を直ちに他の要素制約と因子化することはできない。たとえば、Sit と Sit3 とは因子化不可能なので、Loc1=Loc2 だからと言って (19) と (29) を因子化することはできない。ここで、(29) および (30) からのエネルギーの伝播が起こり、これらに繋がった様々な要素制約が活性化されていると考える。こうして、(18) の活性度が高くなっているとすると、(18) が展開され、sub_sit(Sit1,Sit3) が実行されて、以下のような要素制約ができる。

(34) hold(S,Loc1)
(35) hold(S,Live)

上述の通り、(18) は Sit3 が Sit または Sit1 の部分状況だということであるが、どちらであるかはそれだけではわからない。しかし、もしも Sit3 が Sit の部分状況であるとすれば、上の 2 つの要素制約をそれぞれ (29) および (30) と因子化することにより、ポテンシャルを下げることができる。一方、もし Sit3 が Sit1 の部分状況だとすれば、(29) と (30) のポテンシャルを下げることはできない。従って、前者の方が全体としてポテンシャルが低くなるので、より望ましいことになり、前者が選択される。こうして、DUALS-III は「地球に生きている」という答を返すことができる。(この間、実際には計算の途上でいくつかの新しい述語が定義されたのだが、繁雑なので詳細には触れない。)

この推論過程は、部分情報問題を扱う上で重要な意義を持ついくつかの側面を実現している。第 1 に、これは一種のデフォルト推論になっている。即ち、ポテンシャルをより低くする方の選択肢をデフォルトとして扱っていることになる。DUALS の現在の版が上の例題を処理する場合には、推論速度を上げるため、選択しなかった方の可能性は単に捨てているが、それを捨てずに記憶しておくこともできる。それによって、いわゆる非単調推論などを行なうことも可能である。デフォルト推論は不完全な情報によって判断を下す際に、非単調推論はそうして下した判断を棄却して信念を改変するのに必須である。

第 2 に、活性拡散によって、当面の問題に無関係な知識の部分が正しく無視されている点に注意されたい。このような処理の枠組は、一般的なフレーム問題 [5, 14] に対処する上で不可欠のものである。上記の説明は無関係な知識には触れていないが、たとえテキストの文を大量に読み込んで上記の質問には無関係な知識が大量に蓄えられた後でも、DUALS は、以前とほぼ同じ応答時間で同じ質問に答える。即ち、大量の無関係な知識を単に無視することができる。

第 3 に、上の例において会話の含意 (conversational implicature) が処理されている。つまり、質問応答において、DUALS-III が質問者に情報を提供するのみならず、質問文が暗黙の内に含意している内容が DUALS-III の知識に逆に流れ込んでいる。その含意とは、DUALS-III が質問の答を知っている、というものであり、質問の内容に関する要素制約 (29)~(31) が DUALS の信念に直接付加され、しかも仮説コストが高いという状態に対応する。即ち、この状態は、それらの要素制約を仮説コストの低い他の要素制約と因子化することによってポテンシャルを下げるべきだということを意味し、これは、その質問の答えを導出し得るべきだという会話の含意と見なすことができる。

ここで重要なのは、依存伝播が情報の流れる方向を可能性としては限定していないということである。この性質により、部分情報問題において生ずる多様な情報の流れを実現することができる。たとえば、質問文から情報を得ることを禁じてはいないということによって、上の会話の含意の処理が自然に生じている。このような処理は、従来の手続き型の方法論においては明示的なプログラミングを必要とする。これは、制約プログラミングが談話理解のような部分情報問題を扱う上で見通しのよい方法を与えることを例証している。

5 おわりに

自然言語の処理のような部分情報問題を扱うための依存伝播という制約プログラミングの方法を提示し、言語処理や言語理論に対するその立場からの帰結について論じた。また、その応用例として、談話理解実験システム DUALS-III における問題解決について述べ、情報の部分性を扱うためのさまざまな処理が制約パラダイムによって自然に実現されることを指摘した。

部分情報問題の処理において実現すべき最も重要な処理の側面は、情報の流れの多様性である。言語処理のような部分情報問題においては、莫大な多様性を持つ文脈に応じて、実現されるべき情報の流れもまた莫大であり、予測不能である。適当な文脈を与えればほとんどあらゆる情報の流れが生じ得るようにするため、依存伝播では、一般の導出原理を含むような、なるべく一般的な推論の方式を採用し、さらに活性拡散を用いている。一方、個々の文脈における組合せ的な処理の範囲を小さく限定するため、ヒューリスティックな実行制御を行なっている。

情報の流れの多様性にいかに対処するかに関連するソフトウェア工学上の重要な問題で、本論文では扱うことができなかったものがいくつかある。中でもとりわけ重要な問題は、デバッグの方法に関するものであろう。情報の流れが予測不能であるような問題領域においては、たとえばプログラムの実行過程のトレースを見ても人間には理解し難いなどの事情により、従来のデバッグの方法がそのままでは使えない。これは制約プログラミングの問題と言うよりは、談話処理のような問題を扱う際の情報処理過程そのものの性質による一般的な問題である。

依存伝播のような新しいプログラミング・パラダイムの実際上の有効性は、今後その使用経験を蓄積するにつれて明らかにされるべきである。現在、依存伝播の応用として、談話処理システム全体を制約の統一的な処理として実現しようとしているが、その過程を通じて、この方法の有用性を実証して行きたい。

参考文献

[1] Barwise, J. and Perry, J.: *Situations and Attitudes*, MIT Press, 1983.

[2] Barwise, J. and Etchemendy, J.: *The Liar: An Essay on Truth and Circular Propositions*, MIT Press, 1987.

[3] Charniak, E.: A Neat Theory of Marker Passing, *Proceedings of AAAI'86* 1986.

[4] Colmerauer, A.: *An Introduction to Prolog III*, unpublished manuscript, 1987.

[5] Dennet, D.: The Cognitive Wheels: The Frame Problem of Artificial Intelligence, in *Minds, Machines and Evolution*, Cambridge University Press, 1984. (邦訳: 信原 幸弘: コグニティヴ・ホイール — 人工知能におけるフレーム問題, 現代思想, Vol. 15 (1987).)

[6] Dincbas, M., Simonis, H. and Van Hentenryck, P.: Solving a Cutting-Stock Problem in Constraint Logic Programming, *Proceedings of the 5th International Conference of Logic Programming* (1988), pp. 42-58.

[7] Hanks, S. and McDermott, D.: Nonmonotinic Logic and temporal Projection, *Artificial Intelligence*, Vol. 33 (1987), pp. 379-412.

[8] Hasida, K.: *Bounded Parallelism: A Theory of Linguistic Performance*, Doctoral Dissertation, Univ. of Tôkyô, 1985.

[9] Hasida, K., Ishizaki, S., and Isahara, H.: A Connectionist Approach to the Generation of Abstracts, in Kempen, Gerard (ed.), *Natural Language Generation: New Results in Artificial Intelligence, Psychology, and Linguistics*, Martinus Nijhoff Publishers (Kluwer Academic Publishers) (1987), pp. 149-156.

[10] Hasida, K. and Ishizaki, S.: Dependency Propagation: A Unified Theory of Sentence Comprehension and Generation, *Proceedings of the 10th IJCAI* (1987), pp. 664-670.

[11] 橋田 浩一・白井 英俊: 条件付単一化, コンピュータ・ソフトウェア (1986), Vol. 3, pp. 28-38.

[12] 橋田 浩一: 依存伝播, 第 29 回プログラミング・シンポジウム論文集 (1988), pp.147-158.

[13] Hobbs, J., Stickel, M., Martin, P., and Edwards, D.: Interpretation as Abduction, *Proceedings of the 26th Annual Meeting of ACL* (1988), pp. 95-103.

[14] 松原 仁・橋田 浩一: 人間におけるフレーム問題の解決不能性について, 人工知能学会第 5 回人工知能基礎論研究会 (1989), pp. 21-30.

[15] Mellish, C.: *Computer Interpretation of Natural Language Descriptions*, Ellis Horwood, 1985.

[16] Minsky, M.: *The Society of Mind*, Simon and Schuster, 1985.

[17] Nakashima, H.: Term Description: A Simple Powerful Extension to Prolog Data Structures, *Proceedings of the 9th IJCAI* (1985), pp. 708-710.

[18] Pietrzykowski, T. and Matwin, S.: Exponential Improvement of Efficient Backtracking — A Strategy for Plan-Based Deduction, *Lecture Notes in Computer Science*, Vol. 138 (1982), pp. 223-239, Springer.

[19] Sakai, K. and Sato, Y.: *Boolean Gröbner Bases*, ICOT Technical Memo No. 488, 1988.

[20] Shieber, S.: A Uniform Architecture for Parsing and Generation, *Proceeding of the 12th COLING* (1988), pp. 614-619.

[21] Simon, H.: The Structure of Ill Structured Problems, *Artificial Intelligence* (1973), Vol. 4.

[22] Sugimura, R., Hasida, K., Akasaka, K., Hatano, K., Kubo, Y., Okunishi, T., and Takizuka, T.: A Software Environment for Research into Discourse Understanding Systems, *Proceedings of the FGCS'88* (1988), pp. 285-295.

[23] Sussman, G. and Steele, G., Jr.: Constraints – A Language for Expressing Almost Hierarchical Descriptions, *Artificial Intelligence* (1980), Vol. 14.

[24] Tamaki, H. and Sato, T.: Unfold/Fold Transformation of Logic Programs, *Proceedings of the Second International Conference on Logic Programming* (1984), pp. 127-138.

[25] Tomabechi, H. and Tomita, M.: Application of the Direct Memory Access Paradigm to NL Interfaces, *Proceedings of the 12th COLING* (1988), pp. 661-666.

[26] Tuda, H., Hasida, K., and Sirai, H.: JPSG Parser on Constraint Logic Programming, *Proceedings of the European Chapter of ACL'89* (1989).

[27] Waltz, D., and Pollack, J.: Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation, *Cognitive Science*, Vol. 9 (1985), pp. 51-74.

[28] Ward, N.: Issues in Word Choice, *Proceedings of the 12th COLING* (1988), pp. 726-731.