1B5-3

コネクショングラフを利用した確率推論

Stochastic inference with Connection Graph

後藤 勇樹[*1] 新出 尚之[*2] 藤田 恵[*3]
Yuki Goto Naoyuki Nide Megumi Fujita

[*1]京都大学 数理解析研究所
Research Institute for Mathematical Sciences, Kyoto University

[*2]奈良女子大学研究院自然科学系情報科学領域
Faculty,Division of Natural Sciences, Nara Women's University

[*3]奈良女子大学大学院人間文化研究科
Graduate School of Humanities and Sciences, Nara Women's University

In the recent years,causality is argued with logically,and with stochastically. But there is no strong connection between these two arguments. Now,We propose Bipartite Graph Proof Procedure, which is constructed with Connection Graph in logic, and with Belief Propagation in stochastically inference. Finally, we show how to consider the causality is, using this proposal.

1. はじめに

何か問題が起きた時に、その背景にある事象に因果関係を見出して解決策を考えるのは、日々の暮らしに於いて良くある出来事である。しかし、この時に我々が考えている因果関係とは何かを説明することは、想像以上に難しい問題となる。

因果関係に対する著名な二つのモデル化として、論理を利用した論理プログラムと、確率を利用したベイジアンネットワークが存在し、高度な進化を遂げている。本稿では、論理プログラムに於ける解探索手続きを一般化した Connection Graph Proof Procedure[Kowalski 86][Kowalski 11] を元に、ベイジアンネットワークに於ける厳密推論を行う確率伝播法 [Pearl 88] と似た手法を組み込んだ Bipartite Graph Proof Procedure を提唱し、因果関係についての考察を行う。

2. 準備

以下では通常の一階述語論理上で話を進める。ここでは [Lloyd 87] と [Hindley 97] を参考にした。$u, v, X, Y, ...$ を変数を表す記号とし、$A, B, ...$ を原始式を表す記号、$s, t, ...$ を項を表す記号とする。また、$var(s)$ を項 s 中の変数全体の集合、$var(A)$ を原始式 A 中の変数全体の集合とする。必要に応じて添字を付けた記号も利用する。

ここではプログラム節を、$A : -B_1, B_2, B_3, ..., B_n$ の形をした節で、$n \geq 1$ を満たすものとする。$n = 0$ で原始式中に変数を持たないものを、事実 ($Fact$) と呼ぶ。節中の A は $head$ と呼び、$B_1, B_2, B_3, ..., B_n$ は $body$ と呼ぶ。ゴール節とは、$: -C_1, C_2, C_3, ..., C_m$ の形をした節である。但し、$(m >= 1)$ とする。

以下では、プログラムをプログラム節の集合として表す。また、ゴールを一つのゴール節として表す。この時、プログラムを P、ゴールを G、事実を F の記号で表す。

$\{u_1/s_1, u_2/s_2, u_3/s_3, ..., u_n/s_n\}$ の形のものを代入と呼ぶ。但し、$u_1, ..., u_n$ は互いに異なる。$\theta, \sigma, \tau, ...$ を代入を表す記号とする。$s_1, s_2, s_3, ...s_n$ が全て変数である時、これを純変数代入と呼ぶ。空集合も恒等代入として扱う。恒等代入は ϵ の記号で表す。

項 t、代入 $\theta = \{u_1/s_1, u_2/s_2, u_3/s_3, ..., u_n/s_n\}$ が与えられた時、項 t 中の変数 $u_1, u_2, u_3, ..., u_n$ をそれぞれ $s_1, s_2, s_3, ..., s_n$ に書き換えたものを代入例 $t\theta$ と呼ぶ。原始式 A に対する代入例 $A\theta$ も同様に定義する。また、$dom(\theta) = \{var(u_1), ..., var(u_n)\}, cod(\theta) = \{var(s_1), ..., var(s_n)\}$ と定める。

原始式 A, B が与えられた時、$A\theta = B\theta$ を満たす代入 θ を A と B の単一化子と呼ぶ。A と B の各単一化子 σ に対して $\sigma = \theta\tau$ を満たす代入 τ が存在するような、A と B の単一化子 θ を A と B の最汎単一化子 (most general unifier) と呼ぶ。以下では、最汎単一化子の事を $m.g.u.$ と略記する。A と B の間に $m.g.u.$ が存在する時、A と B は単一化可能であると言う。

原始式 A, B が与えられた時、$A\theta = B\sigma$ を満たす代入 θ, σ が存在する時、この $A\theta$ を A と B の common instance と呼ぶ。A と B の全ての common instance D に対してある代入 θ が存在し $D = C\theta$ を満たすような、A と B の common instance C を most general common instance と呼ぶ。以下ではこれを $m.g.c.i.$ と略記する。

プログラム節 $A : -B_1, B_2, B_3, ..., B_n$ と $C : -D_1, D_2, D_3, ...D_m$ が与えられた時、B_i と C が $m.g.u.\theta$ によって単一化可能である場合、二つのプログラム節の resolvent を $A : -B_1, B_2, ..., B_{i-1}, D_1\theta, ...D_m\theta, B_{i+1}, ...B_n$ で定める。

以下二つの定義は、本論文の技術的な要請からくるものである。今、$\theta = \{u_1/s_1, ..., u_n/s_n\}, \sigma = \{v_1/t_1, ..., v_m/t_m\}$ という二つの代入を考える。$\theta \circ \sigma$ と $\theta \diamond \sigma$ を以下で定める。

$$\begin{aligned}
\theta \circ \sigma &= \{u_i/t_j \mid \exists j.u_i = v_j\} \\
&\cup \{u_i/s_i\tau \mid \text{上記以外}\} \\
\theta \diamond \sigma &= \{u_i/s_i \mid u_i \notin \{v_1, ..., v_m\}\} \\
&\cup \{v_i/t_i \mid v_i \notin \{u_1, ..., u_n\}\} \\
&\cup \{u_i/s_i \mid \exists j.u_i = v_j \wedge s_i = t_j\}
\end{aligned}$$

連絡先: 後藤 勇樹, 京都大学数理解析研究所, 〒 606-8502, 京都市左京区北白川追分町,FAX: 075-753-7272,gotoyuki@kurims.kyoto-u.ac.jp

但し $\theta \circ \sigma$ 中の τ は、$\{u_1, ..., u_n\} \cap \{v_1, ...v_m\} = \{x_1, ...x_l\}$ とする時、$x_i\theta\tau = x_i\sigma\tau$ を全ての i について同時に満たすものとする。このような τ が無ければ、$\theta \circ \sigma$ は未定義とする。また、$\theta \diamond \sigma$ は、$\exists < i, j > .v_i = u_j \wedge s_i \neq t_j$ の時未定義とする。$\circ, \diamond$ 共に二項演算として考える時、右結合とする。

3. Bipartite Graph Proof Procedure

この章では、今回提案する Bipartite Graph Proof Procedure の定義を述べていく。今、プログラム P, 事実 F, ゴール G が与えられているとする。以下では、プログラム全体 $\mathcal{P}$ を $\mathcal{P} = P \cup F \cup G$ で定義する。

以下では先ず、プログラム全体から、部分的にラベル付けされた二部有向グラフを構成し、次にこのグラフを利用してゴールを解く正解代入を探し出す。

3.1 二部有向グラフの構成

今、プログラム全体 $\mathcal{P}$ が与えられているものとする。下準備として、$\mathcal{P}$ 中の各々のプログラム節中の変数を変更し、同じ変数が二つ以上のプログラム節で使われないようにしておく。また、同一のプログラム節 2 つで単一化可能である場合、そのプログラム節の複製を上記の変数の制約条件を満たすように追加する。

さて、プログラムから二部グラフ $\mathcal{G} =< H, B, A, R >$ を以下の方法で構成する。

- H:$\mathcal{P}$ 中の各節の $head$ に対応するノード
- B:$\mathcal{P}$ 中の各節の $body$ にある原始式に対応するノード
- A:$\mathcal{P}$ 中の各節の $body$ 部に対応する各ノードから $head$ に対応するノードへの有向エッジ
- R:単一化可能な二つの節に対し、単一化される $head$ に対応するノードから $body$ 部に対応するノードへの有向エッジにラベルとしてその $m.g.u.\theta$ を付けたもの。但し、$cod(\theta)$ は $head$ に現れる変数のみの集合にする。

例として、以下で示すプログラムに対応するグラフを図 1 に示す。この例では、H に属するノードを ■ で表し、B に属するノードを ◯ で表している。

```
grandparent(X,Z):-parent(X,Y),parent(Y,Z).
parent(A,B):-mother(A,B).
parent(C,D):-father(C,D).
```

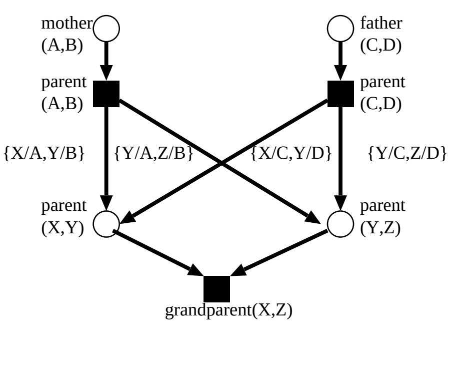

図 1: 二部グラフ構成の具体例

3.2 単一化子伝播

以下では、単一化子をメッセージとしてノード間で送信し、正解代入を求める方法を解説する。解説の為に、幾つかの定義を導入する。有向グラフ上の葉ノードとは、自分自身に向けた有向エッジが存在しないノードを指す。また、あるノードの親ノードとは、そのノードに向けた各々の有向エッジの反対側にあるノードを指す事とする。

- $h \in H$ を葉ノードとし、$b \in B$ とラベル θ 付き有向エッジ $r \in R$ で結ばれているとする。今、h から b へのメッセージを、$\mu_{h \to b} = \theta$ で定める。
- $b \in B$ を葉ノードとし、$h \in H$ と有向エッジ $a \in A$ で結ばれているとする。今、b から h へのメッセージを、$\mu_{b_i \to h} =!$ で定める。

B に属する葉ノードからのメッセージとしてここで定めた ! は、そこで解探索に失敗する事を意味している。以上二つを図 2 に示す。

- ノード $b \in B$ について、親ノード全体の集合を $\{h_i\}_{i=1}^n \subset H$ とおく。ノード b が一つ以上の親ノードからメッセージ $\{\mu_{h_{i_j} \to b}\}_{j=1}^m$ を受け取ったとする。この時、b から有向エッジ $a \in A$ で結ばれたノード $h \in H$ へのメッセージは、$\mu_{b \to h} = \cup_{j=1}^m \mu_{h_{i_j} \to b} \upharpoonright var(b)$ で定める。但し、上式の右辺が空集合になった場合、メッセージを送信しない。
- ノード $h \in H$ について、親ノード全体の集合を $\{b_i\}_{i=1}^n \subset B$ とおく。ノード h が全ての親ノードからメッセージ $\{\mu_{b_i \to h}\}_{i=1}^n$ を受け取ったとする。この時、h からラベル θ 付き有向エッジ $r \in R$ で結ばれたノード $b \in B$ へのメッセージは、$\mu_{h \to b} = \{\theta \circ \theta_1 \diamond \theta_2 ... \diamond \theta_n \mid \theta_i \in \mu_{b_i \to h}\} \upharpoonright var(h)$ で定める。但し、上式の右辺が空集合になった場合はメッセージを送信しない。

上の一つ目は、解探索に相当する。メッセージは正解代入に相当するので、一つあれば十分である。上の二つ目では、$body$ 部の全ての式に解が存在する時にメッセージを構成している。構成方法の妥当性は次の章で議論する。

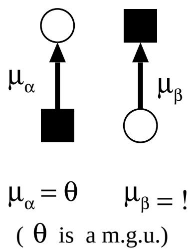

図 2: 初期伝播

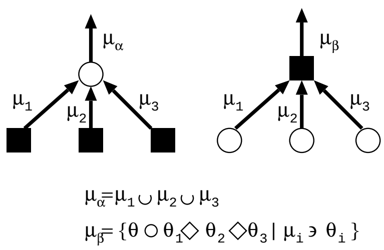

図 3: 再帰的伝播

4. 健全性証明

この章では、以下の定理を証明する。以下では、プログラム全体 $\mathcal{P} =< P, F, G >$ を与えておく。

定理 1 (健全性) ゴール節を : $-C_1, ..., C_n$、各 C_i に送られてきたメッセージを μ_i とおく。今、$\theta_i \in \mu_i$ を定める時、$\theta_1 \diamond ... \diamond \theta_n$ はもし定義されれば、$\mathcal{P}$ の正解代入である。

この定理は以下二つの補題から示される。補題はメッセージ伝播の各ステップで健全性が保たれている事を意味している。

補題 1 $A : -B_1, ..., B_n$ を P 中のプログラム節とし、θ_i を代入とする。今、各 i に対して $B_i\theta_i$ が $P \cup F$ の論理的帰結である時、$A\theta_1 \diamond ... \diamond \theta_n$ も $P \cup F$ の論理的帰結である。

この補題は、以下の命題から証明できる。

命題 1 各 i について代入 σ が存在し、$\theta_1 \diamond ... \diamond \theta_n = \theta_i\sigma$

(補題 1 の証明)　$\theta = \theta_1 \diamond ... \diamond \theta_n$ とおく。今、命題 1 と仮定より各 i に対して $B_i\theta$ は $P \cup F$ の論理的帰結である。故に $A\theta$ は $P \cup F$ の論理的帰結である。□

以下の補題と命題では、A, B を $var(A) \cap var(B) = \phi$ を満たす原始式とする。また、σ を $var(\sigma) \cap var(B) = \phi$ と $cod(\sigma) = \phi$ を満たす代入、θ を A と B の $m.g.u.$ とする。

補題 2 $A\sigma$ が $P \cup F$ の論理的帰結の時、$B(\theta \circ \sigma)$ も $P \cup F$ の論理的帰結である。

命題 2 $A(\theta \circ \sigma) = B(\theta \circ \sigma) = A\sigma$

(命題 2 の証明) θ_1, θ_2 を $dom(\theta_1) = var(A), dom(\theta) = var(B), \theta = \theta_1 \diamond \theta_2$ を満たすように定める。これらは θ が A と B の $m.g.u.$ である為存在する。また、$A\theta_1 = B\theta_2$ である事に注意する。今、τ を $\theta \circ \sigma$ 中で定義された代入とする。定義より、

$$\begin{aligned} A(\theta \circ \sigma) &= A\sigma \\ &= A\theta_1\tau \\ B(\theta \circ \sigma) &= B\theta_2\tau \\ \therefore A(\theta \circ \sigma) &= B(\theta \circ \sigma) \end{aligned}$$

故に示された。□
(補題 2 の証明) 命題 2 より明らか。□

(定理 1 の証明) ゴール節が $: -C$ の時にのみ示せば良い。もし、C に送られたメッセージが葉ノードから来たメッセージであれば、葉ノードは明らかに $P \cup F$ の論理的帰結である為成り立つ。そこで、C に送られたメッセージがプログラム節 $A : -B_1, ..., B_m$ 中の $head$ に対応するノードから来たものとする。今、帰納的仮定として B_i に送られてきたメッセージ θ_i に対して、$B_i\theta_i$ が $P \cup F$ の論理的帰結である事を用いてよい。この時 C と A の $m.g.u.$ を θ とおくと、二つの補題から $C\theta \circ \theta_1 \diamond ... \diamond \theta_m$ が $P \cup F$ の論理的帰結である事が言える。以上より、C に入ってくる全てのメッセージに対して、その元は $\mathcal{P}$ の正解代入である事が示された。□

5. 例と確率伝播法との比較

本稿で述べた Bipartite Graph Proof Procedure は、ベイジアンネットワークにおける確率伝播法を元に構成している。この章では具体例を通じて、その相違について述べる。また、こうした比較を元に、因果関係に対する我々の所見を述べる。
図 4(a) に示すのは、以下のプログラムを今回の手法で図式化したものである。

```
Goal                Program
  :-car_move.         car_move :-
Fact                       oil_is_satisfied,
  drive.                   drive.
  oil(75).            oil_is_satisfied :-
                           oil(X),
                           X>=70.
```

これに対し、図 4(b) に、このプログラムをベイジアンネットワークで表現したものを載せておいた。ここで X, Y, Z は、確率変数を表す。この図は Goal と Fact のみをノードに持っているが、内部の条件付き確率でプログラム部分を再現できる。ここでは詳細を述べないが、ベイジアンネットワークの確率伝播法は、因子グラフと呼ばれる二部グラフを構成し、この上で確率伝播アルゴリズムを行う。詳細については [Bishop 07] を参照して欲しい。
今回の場合、因子グラフは図 4(c) となる。図中に書かれている $f_a(X)$ 等は因子を表す。このグラフでは、$f_a(X) = p(X), f_b(Y) = p(Y), f_c(X, Y, Z) = p(Z|X, Y)$ となる。

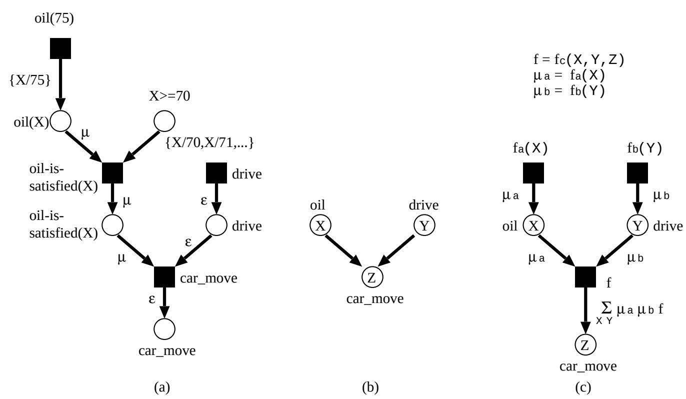

図 4: 比較

ここで注目すべき点が二つある。一つは (c) より (a) の方が、ノードの数が多いことである。もう一つは、(a) における条件節 $X \geq 75$ が、(c) においては因子 $f_c(X, Y, Z)$ の内部で確率値の条件分岐として与えられる事である。
これらの事から、論理的な因果関係は確率的に書く因果関係よりも詳しく記述する必要があると言える。逆に言えば、背景にある論理的な因果関係が正確に分からない時も、確率的な因果関係を用いれば、十分実用に耐えうるものを作れるのではないかと思われる。ただ、我々はこの議論に必要な定性的な方法を、現段階では持ち合わせていない。

6. コネクショングラフとの関係

コネクショングラフは、[Kowalski 75] で R.Kowalski によって提唱された概念である。推論、演繹、発見といった論理プログラムの重要な概念は、全て resolution で説明出来るのだが、コネクショングラフでは resolution を主軸においた procedure を構成する事で、これを一つのグラフ還元の方法に纏めている。

6.1 Connection Graph Proof Procedure

今、プログラム P, 事実 F, ゴール G が与えられているとする。この時、プログラム全体を以前と同様に $\mathcal{P} = P \cup F \cup G$ で定めておく。
以下で、Initial Graph $\mathcal{G} =< N, E >$ を構成する。

- N：$\mathcal{P}$ に属する各節中の原始式に対応するノード
- E：$\mathcal{P}$ に属する二つの単一化可能な節について、単一化される原始式に対応するノード間を $m.g.u.$ でラベル付けして結び付ける無向リンク

構成された Initial Graph を元に以下の操作を繰り返す。最終的にノードが全てなくなった時に、ゴールは達成される。

- リンクが張られていないノードが存在する時、そのノードが属していた節中の原始式に対応する全てのノードを除去する。また、関連するリンクも除去する。
- トートロジーが存在する場合、それを除去する。また、関連するリンクも除去する。
- リンクを一つ選択し、それを除去する。その後、リンクで結ばれた二つの節の resolvent をグラフに追加し、単一化可能な節を見付けて $m.g.u.$ でラベル付けしてリンクを貼る。

図 5 で、Initial Graph と操作の例を示す。

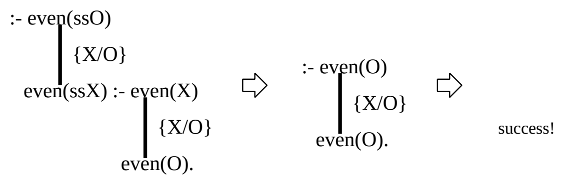

図 5: プロセスの一例

6.2 問題点と解決策

Connection Graph Proof Procedure には、幾つか複雑なプロセスが存在する。例えば以下のプログラムの様に相互参照する形のトートロジーが存在する為、この除去には注意を要する。

```
even(X):-odd(sX).
odd(sY):-even(Y).
```

今回提唱した方法では、トートロジーになる部分には恒等代入のメッセージが送られる為、特別な扱いをする必要がない。図 6 で、トートロジーが今回の手法でどのように扱われているかを示す。また、コネクショングラフのプロセスでは、新たにグラフが更新される度に、単一化子がどのように変更されるのかは議論しにくい。少し複雑な定義にはなっているが、今回の手法では新しい単一化子を構成する演算を示せている。

逆に、今回の手法がコネクショングラフに及んでいない所も多い。特にメッセージの構成方法は、コネクショングラフとは違い bottom-up 的な手法のみしか扱えていない。今回のグラフを用いて top-down 的なメッセージの伝播方法を構成できれば、因果関係に対するより深い理解に貢献できるのではないかと思われる。

7. まとめと将来展望

確率概念を入れて論理プログラムを拡張する事によって、ベイジアンネットワークを論理プログラムの中で扱う研究が、統計的アブダクションという形で知られている。[佐藤 10] しかし、ベイジアンネットワークの手法を用いて論理プログラムを

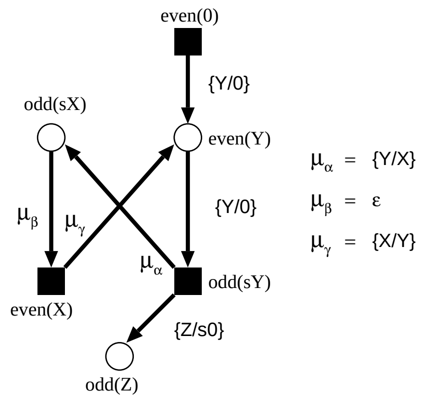

図 6: トートロジーについて

扱うという逆方向の研究は、これまで行われて来なかったのではないだろうか。

今回の研究は、両者に共通する因果関係という一つの概念に着目し、論理と確率に対する新たな関係性を示唆してみた。ただ、今回は提唱した手法の完全性を示す事が出来なかったのが気がかりであり、今後の課題として解決したいと思っている。

将来展望として、d-分離、因果的マルコフ条件といったベイジアンネットワークの概念を、論理プログラミングの中で活かす事が出来たら面白いのではないかと考えている。

参考文献

[Bishop 07] Bishop, C. M.: *Pattern Recognition and Machine Learning*, Springer, first edition (2007), (元田, 栗田, 樋口, 松本, 村田 (監訳). パターン認識と機械学習 (上・下), シュプリンガー・ジャパン, 2008)

[Hindley 97] Hindley, J. R.: *Basic simple type theory*, Cambridge University Press, New York, NY, USA (1997)

[Kowalski 75] Kowalski, R.: A Proof Procedure Using Connection Graphs, *Journal of the Associtaion for Computing Machinery*, Vol. 22, No. 4, pp. 572–595 (1975)

[Kowalski 86] Kowalski, R.: *Logic for problem-solving*, North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands (1986)

[Kowalski 11] Kowalski, R. A.: *Computational logic and human thinking : how to be artificially intelligent*, Cambridge University Press, Cambridge (2011)

[Lloyd 87] Lloyd, J.: 論理プログラミングの基礎, ソフトウェアサイエンスシリーズ, 産業図書 (1987)

[Pearl 88] Pearl, J.: *Probabilistic reasoning in intelligent systems: networks of plausible inference*, The Morgan Kaufmann series in representation and reasoning (1988)

[佐藤 10] 佐藤 泰介：統計的アブダクション (〈特集〉論理に基づく推論研究の動向), 人工知能学会誌, Vol. 25, No. 3, pp. 400–407 (2010)