■S3 群（脳・知能・人間）－ 3 編（人工知能と学習）

2 章 知識と推論

（章主任：岩沼宏司）［2018 年 12 月 受領］

■概要■

知識の表現と推論は，人工知能研究の初期からの基本的で重要な中心的研究課題である．知識は古くは絵や口頭伝承によって伝えられてきた．文字の発明以降は，図や絵をまじえた文章により，より多くの知識が，より正確に，より多くの人々に，長い時間と広い空間を超えて，伝えられるようになってきた．印刷と製紙技術の発展に伴い，文書の重要性は急激に増加し，近年のインターネットと WEB 技術の発展により，図や絵をまじえた文書の重要性は益々増してきている．

人工知能における知識の表現と推論の研究は，一言で言えば，図や絵を含めた文章による知識表現とその上の推論の科学的理解と数理化，及びコンピュータ上での工学的実現を目標としてきた．一口に，知識の表現と推論と言っても多種多様なものが存在するために，人工知能における研究も様々なものがある．また，近年の科学技術の発展は目覚ましく，新しい知見，例えば生物学的における知見などが人工知能の各種の研究へ影響を及ぼしてきた歴史もある．本章では，これまでの知識表現と推論に関する研究から代表的なものを取り上げて，基本的な解説を行う．

【本章の構成】

知識は，古くから図や絵をまじえた文章によって，時間と空間を超えて多くの人々に伝えられてきた．この図や絵や文章による知識表現とその上の推論は，人工知能研究の初期から基本的かつ重要な課題として，実に様々な研究が行われてきた．以下の節では，幾つかの代表的な手法を取り上げて基本的な解説を行う．

2-1 節では，問題解決のための推論，特に探索とプランニング技術について解説を行う．これはロボットや工作機械のプランニングなどに必須の理論と技術である．探索やプランニングの対象となる物理空間あるいは論理的空間の構造の情報や制約，すなわち知識は，グラフの形式で表現されることが多い．また，推論は，グラフ中の制約条件を満たす最短経路などの計算として定式化されることが多い．これらの基本的事項について解説される．

続く 2-2 節は，伝統的な知識表現と推論についての解説である．ここでは，知識は複数の文章で記述され，それぞれの文章は複数の単語を要素とする構造を持っていると考える．文章や単語は相互に様々な形で結合して関係を持っている．知識は非常に複雑な構造で表現されているが，これらを考察・整理して提案された知識表現の形式として，意味ネットワークやフレーム表現などがある．複数の知識源を統合して推論を行う方式・枠組みとしては黒板モデルがあり，規則の形をした知識に基づく推論システムとしてのプロダクションモデルなどがある．これらの基本について解説を行う．

2-3 節では，論理に基づく知識表現と推論の解説される．数理論理学では，論理式と呼ばれる記号列によって知識・性質を表現し，その記号列変換によって，常に正しい（恒真）式を導出する．コンピュータの基本機構と親和性が非常に高いため，コンピュータ科学の重要な基盤

となっている．最も基本的な論理体系は命題論理であるが，その応用としては，大量の知識やデータを効果的に格納・利用するためのデータ構造 ZDD/BDD や，多くの制約充足問題を表現し解決できる SAT 問題とその高速ソルバーがあり，その基本的な解説を行う．次に述語論理の応用としての論理プログラミングを取り上げ，不完全な知識を扱うための「失敗としての否定」規則や，不確定な知識を表現するための選言的プログラミング，及び両者の統合形態である解集合プログラミングなどについて基本的な解説を行う．

次の 2-4 節は，制約に基づく推論についての解説である．ここで制約とは，幾つかの変数が同時にとることができる値の組合せを示す関係であり，これは宣言的な知識表現である．その上の推論とは，すべての制約を充足する変数への値の割当てを求めることである．具体的な推論は，制約の簡約化と解空間の探索が基本となる．現実の多くの数理的問題が制約充足問題として定式化できることから，実用上もとても重要である．

2-5 節は，不確実性を扱う知識表現と推論に関する解説である．現実のデータや事象には，必ず観測誤差や揺らぎが混入することから，不確実性を持つ知識やデータを適切に表現し，その上で合理的に推論することは，実用上，極めて重要である．確率理論は，この不確実性を取り扱う最も代表的な手法である．ここでは，確率に基づく推論の代表例であるはナイーブベイズ法や，ベイジアンネットワークなどを紹介する．また，従来の宣言的知識での不確実性の取り扱いや，確率推論との融合などについて基本的な解説を行う．

最後の 2-6 節は，生命にならう知識表現と推論に関する解説である．生命にならうアプローチの代表的なものは，進化的計算と称される一連の手法である．最も歴史が古いものは遺伝的アルゴリズムと呼ばれているもので，遺伝子を模した複数の記号列で知識やデータを表現する．推論は最適化計算を行っており，複数の記号列を用いた遺伝子操作，すなわち交叉や突然変異を模した計算を行う．また適宜，記号列（遺伝子）の目標（環境）に対する評価と，それに基づく選択淘汰を行って最適解を探していく．本節では，遺伝的アルゴリズムのほかにも進化的プログラミングや群知能などについて基本的な解説を行う．

2-1 問題解決のための推論：探索とプランニング
2-2 伝統的な知識表現と推論
2-3 論理に基づく知識表現と推論
2-4 制約に基づく推論
2-5 不確実性を扱う知識表現と推論
2-6 生命にならう知識表現と推論

■S3 群－3 編－2 章

2-1 問題解決のための推論：探索とプランニング

（執筆者：北村泰彦）［2011 年 3 月 受領］

人工知能が扱う問題解決の多くはその手順が明確ではなく，試行錯誤を行いながら解を発見する場合が多く，その過程は状態空間における探索（Search）として定式化されている．また，ロボットなどの動作を計画するプランニングも目標を達成するための動作の系列を求める探索問題の一つとみなすことができる．本節では人工知能で用いられる様々な探索手法について解説する．

2-1-1 状態空間

状態空間（State Space）は，状態（State）の集合と状態を変化させるオペレータ（Operator）の集合により定義され，グラフとして表現される．問題は，状態空間，初期状態（Initial State），目標状態（Goal State）により与えられ，問題解決とは初期状態を目標状態に変化させるオペレータの系列，すなわち初期状態から目標状態までの経路を求めることである．

2-1-2 力ずく探索手法

最も単純な探索手法のカテゴリーは，与えられた状態空間の情報のみを用いるもので，力ずく探索（Brute-force Search），情報なし探索（Uninformed Search），盲目的探索（Blind Search）などと呼ばれる．問題固有の知識や情報を用いないので，人工知能以外の分野でも利用される汎用的な探索手法である．

(1) 幅優先探索と均一コスト探索

幅優先探索（Breadth-first Search）（あるいは横型探索）は，目標状態が見つかるまで，初期状態に近いものから，**図 1･1** に示す順序で状態を展開していく手法であり，目標状態までの最短経路を発見することができる．展開とは，ある状態に適用可能なすべてのオペレータを適用して，すべての継続状態を求めることである．

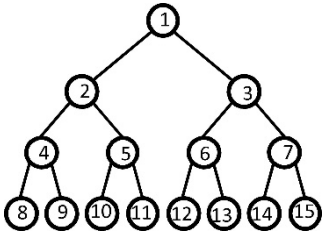

図 1・1 幅優先探索

状態空間グラフの分岐係数を b，目標状態の深さを d の木と仮定した場合，幅優先探索の時間計算量は，最悪で$b + b^2 + b^3 + \cdots + b^d = O(b^d)$となる．幅優先探索の欠点は探索に必要な記憶量である．初期状態から近い順に状態を展開していくためには，展開の候補となる状態をすべて記憶しておく必要があるので，空間計算量も$O(b^d)$となる．したがって，幅優先探索では，コンピュータの記憶容量が制約となって大きな状態空間の探索は難しい．

オペレータ適用のコストが一定でない場合は，幅優先探索は均一コスト探索（Uniform-cost Search）に一般化される．この場合，状態の展開は初期状態から近い順ではなく，初期状態からの経路のコスト順になる．均一コスト探索では最小コスト経路を発見することが保証され，ダイクストラ法（Dijkstra's Algorithm）とも呼ばれている．

(2) 深さ優先探索

深さ優先探索（Depth-first Search）（あるいは縦型探索）は，幅優先探索とは対照的に，**図 1・2** に示すように，初期状態から始めて，より深い状態へと順に展開していく手法である．深さ優先探索の利点は必要な記憶量であり，初期状態から展開する状態までの経路のみを記憶しておけばよいので，探索の深さに対して線形に増加するだけである．時間計算量は，最悪の場合，展開する状態の数は幅優先探索と変わらないので，$O(b^d)$である．

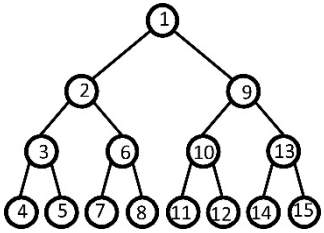

図 1・2 深さ優先探索

深さ優先探索の欠点は状態空間が無限の大きさを持つ場合，展開を無限に繰り返してしまう可能性があることである．そこで，探索の深さを制限することが考えられるが，目標状態の深さを d としたとき，深さ制限を d より小さくすると解を発見することができなくなる．また，d より大きくすると無駄な探索が多くなり，最短でない経路が得られる場合がある．

(3) 深さ優先反復深化探索

深さ優先反復深化探索（Depth-first Iterative-deepening Search）は，深さ優先探索の欠点を改良したもので，深さ制限を徐々に深くしながら，深さ優先探索を繰り返す手法である．すなわち，**図 1・3** に示すように，深さ制限を最初は 1 にして深さ優先探索を行う．目標状態が発見できなかった場合は，深さ制限を 1 増加させて再び深さ優先探索を行う．これを目標状態が発見されるまで繰り返す．徐々に制限を深くしていくので，最短経路の発見は保証される．空間計算量は深さ d の目標状態を発見すれば終了するので，$O(d)$となる．

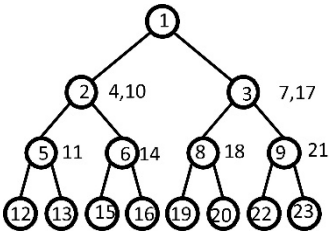

図 1・3 深さ優先反復深化探索

一方，深さ優先反復深化探索は同じ状態を何度も展開することになり，無駄な探索が多いが，時間計算量は幅優先探索と同等である．なぜなら，深さ d の状態の数はb^dで，これは 1 回だけ生成される．深さ $d-1$の状態の数は b^{d-1}であり，2 回生成される．以下同様に考えると，生成される状態の数は$b^d + 2b^{d-1} + 3b^{d-2} + \cdots + db$であり，$b > 1$ なので，$O(b^d)$である．

(4) 双方向探索

双方向探索（Bidirectional Search）（あるいは両方向探索）は目標状態がユニークに定義されている場合，初期状態から目標状態への探索だけでなく，目標状態から初期状態への探索も並行して行う手法である．2 つの探索が状態空間のいずれかの状態でつながれば解が発見されたことになる．双方向探索では最短経路の発見が保証されている．双方向探索の時間計算量は解経路長の半分まで探索すればよいので，時間計算量は $O(b^{d/2})$ である．2 つの探索の共通状態を求めるには少なくとも一方の探索は幅優先探索を行う必要があり，空間計算量は $O(b^{d/2})$ である．

2-1-3 ヒューリスティック探索

力ずく探索では状態空間に明示的に表現されている情報のみを利用して探索が行われている．しかし，状態空間が大きくなると探索に必要な計算量や記憶量が急激に増大する，組合せ爆発（Combinatorial Explosion）が生じる．そこで，人工知能においては問題領域に特化した知識を利用して探索の効率を上げる手法が研究されてきた．このような知識のことをヒューリスティックス（Heuristics）あるいは発見的知識と呼ぶ．

(1) ヒューリスティック関数と最良優先探索

状態空間探索で利用されるヒューリスティックスの代表例に目標状態までの直線距離を表すユークリッド距離（Euclidean Distance）や，各座標の差の絶対値の和であるマンハッタン距離（Manhattan Distance）などがある．ヒューリスティックスが満たすべき性質は，それが最小経路コストの推測値となりうることと，その計算が容易であることが挙げられる．

状態 n から目標状態までの経路の最小コスト推測値をヒューリスティック関数 $h(n)$ で表す．ヒューリスティック関数をそのまま評価値として探索を行う手法として最良優先探索（Best-first Search）がある．本手法では，既に展開した状態を記録する Closed リストと展開の候補となる状態を保存する Open リストを用いる．探索は Open リストに初期状態を入れることにより開始され，Open リスト内の $h(n)$ が最小の状態 n を展開して，Closed リストに入れる．展開して得られた継続状態は h の値を計算して，Closed リストに含まれていなければ，Open リストに挿入する．探索は展開の候補として目標状態が得られた時点で終了する．この手法の欠点は展開の際に $h(n)$ のみを考慮し，初期状態から n までのコストを考慮しないので，最小コスト経路の発見が保証されない点である．

(2) A*アルゴリズム

初期状態から状態 n を経由して，目標状態まで至る経路の最小コスト推測値を $f(n) = g(n) + h(n)$で表す．ここで $g(n)$は初期状態から状態 n までの経路の最小コスト推測値である．A*アルゴリズム（以下 A*）は $f(n)$を評価値として Open リストから展開する状態を選択する．A*は ヒューリスティック関数 $h(n)$が実際のコストを上回らなければ最小コスト経路の発見が保証される．一方，A*の欠点は探索に必要な記憶量であり，幅優先探索と同様に，展開する状態の候補をすべて Open リストに記憶しておく必要がある．

(3) 反復深化 A*アルゴリズム

深さ優先反復深化探索が力ずく探索における記憶量の問題を緩和したように，反復深化 A*（Iterative Deepening A*）は A*の記憶量の問題を緩和する．本手法は，探索する経路コストの上限を $f(n) = g(n) + h(n)$ に制限して，深さ優先探索を繰り返す．まず，初期状態の評価値を上限として探索を行う．目標状態が発見されなかった場合は，展開された状態の評価値の中で前回の上限以上で最小のものを新たな上限として探索を繰り返す．ヒューリスティック関数 $h(n)$ が実際の最小経路コストを上回ることがなければ，反復深化 A*は最小コスト経路を発見し，必要な記憶量は探索の深さに対して線形である．

2-1-4 プランニング

プランニング（Planning）とは，あるタスクを達成するための，ロボットなどによる行動の系列を求める手法である．プランニングも，初期状態，状態を変化させるオペレータの集合，達成すべき目標状態から構成される状態空間探索として定式化できる．プランニングにおける目標状態は，副目標の AND 表現で表されるような複雑な場合もある．したがって，代表的なプランニングシステムである STRIPS では目標状態から初期状態に至る後ろ向き探索が行われる．目標状態が複数の副目標から成り立っている場合，片方の副目標の達成が他方を副目標の達成を妨げる場合があり，これらの干渉をどのように扱うかがプランニングでは重要になる．

プランニングの例題としてブロックワールドが広く用いられている．ブロックワールドは，ロボットハンド，テーブルと幾つかのブロックから構成されており，それぞれの状態は 5 つのリテラルによって表現できる．

- ONTABLE(x)：ブロック x はテーブルの上にある．
- ON(x,y)：ブロック x はブロック y の上にある．
- CLEAR(x)：ブロック x の上には何も置かれていない．
- HOLD(x)：ロボットはブロック x をつかんでいる．
- EMPTY：ロボットは何もつかんでいない．

オペレータは以下のように表現される．オペレータには，オペレータが適用可能な前提条件と，適用後の事後状態が削除リストと追加リストによって記述されている．

- PICKUP(x)：テーブルの上にあるブロック x をつかむ．
  前提条件：ONTABLE(x), CLEAR(x), EMPTY
  削除リスト：ONTABLE(x), CLEAR(x), EMPTY
  追加リスト：HOLD(x)

- PUTDOWN(x)：ブロック x をテーブルの上に置く．
  前提条件：HOLD(x)
  削除リスト：HOLD(x)
  追加リスト：ONTABLE(x), CLEAR(x), EMPTY

- UNSTACK(x,y)：ブロック y 上のブロック x をつかむ.
  前提条件：ON(x,y), CLEAR(x), EMPTY
  削除リスト：ON(x,y), CLEAR(x), EMPTY
  追加リスト：HOLD(x), CLEAR(y)

- STACK(x,y)：ブロック x をブロック y の上に置く.
  前提条件：HOLD(x), CLEAR(y)
  削除リスト：HOLD(x), CLEAR(y)
  追加リスト：ON(x,y), CLEAR(x), EMPTY

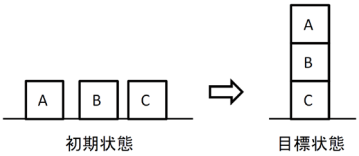

図 1・4　ブロックワールド

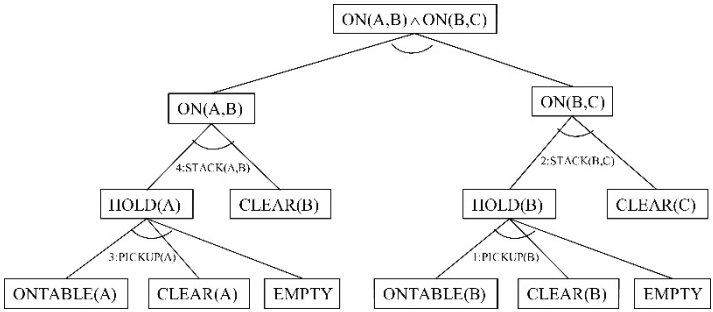

図 1・5　解を示す AND 木

例えば，**図 1・4** に示すように，初期状態が ONTABLE(A), ONTABLE(B), ONTABLE(C), CLEAR(A), CLEAR(B), CLEAR(C)，目標状態が ON(A,B) ∧ ON(B,C)で表されるとする．2 つの副目標を独立とみなして得られた解は**図 1・5** のようになる．まず，目標状態が ON(A,B)と ON(B,C)に分解される．ON(A,B)を達成するためには，それが追加リストに含まれるオペレータ STACK(A,B)を適用すればよい．オペレータ STACK(A,B)が適用可能になるためには，その前提条件を満たす必要があり，それが新たな副目標 HOLD(A)と CLEAR(B)になる．CLEAR(B)は初期状態で既に満たされているので，HOLD(A)が次に達成すべき副目標となり，オペレータ PICKUP(A)が適用される．同様にもう一つの副目標 ON(B,C)に関してもオペレータの系列 PICKUP(B), STACK(B,C)が得られる．

しかし，図 1･5 に示す解グラフでは ON(A,B)と ON(B,C)が干渉している．ON(A,B)を達成するための STACK(A,B)の前提条件は CLEAR(B)であるが，これは HOLD(B)を達成するための

PICKUP(B)の削除リストに含まれており，互いに干渉している．CLEAR(B)は STACK(B,C)の追加リストに含まれるので，それまで STACK(A,B)の実行を延期すればよく，最終的なオペレータの系列は PICKUP(B), STACK(B,C), PICKUP(A), STACK(A,B)となる．

■参考文献

1) 人工知能学会(編)：“人工知能学事典,” 共立出版, 2005.
2) R.E. Korf : “Search,” In S.C. Shapiro (ed.) : “Encyclopedia of Artificial Intelligence,” Wiley, pp.1460-1467, 1995.
3) 白井良明, 辻井潤一：“人工知能,” 岩波書店, 1982.
4) J. Pearl：“Heuristics: Intelligent Search Strategies for Computer Problem Solving,” Addison-Wesley, 1984.
5) R.E. Fikes and N.J. Nilsson：“STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving,” Artificial Intelligence, 2, pp.189-208, 1971.

■S3 群－3 編－2 章

2-2 伝統的な知識表現と推論

（執筆者：市瀬龍太郎）［2008 年 10 月 受領］

人工知能の分野では，長い期間にわたり，様々な知識をどのようにして表現するかが研究されてきた. 本節では伝統的に使われている 5 つの手法を取り上げ, それぞれに関して概説する.

2-2-1 意味ネットワーク

意味ネットワーク（Semantic Network）は，Quillian によって提唱された知識を表現する枠組みである．意味ネットワークでは，知識を表現するための様々な概念や，概念同士の関係をグラフの形式を利用して表現する．概念は，グラフのノードとして表され，概念間の関係は，ノードを結ぶリンクで表される．**図 2･1** は，意味ネットワークの例である．この例では，動物，魚，鳥などの概念がグラフ中のノードとして，記述されており，それぞれの関係がリンクによって記述されている．例えば，鳥という概念は動物という概念のなかの一つであるので，その関係が Is_A という関係により表記されている．これは，「鳥 is a 動物」という 2 つの間の関係を表したものである．同様に，鳥は，羽を持っているので，両者は Has_A という関係によって表記されている．図 2･1 では，関係として，Is_A と Has_A の 2 種類しか提示していないが，実際の意味ネットワークでは，性質を表す関係など，様々の記述を加えることによって，多様な知識を表現することが可能となる．

意味ネットワークで表された知識を用いると，図が直接示している知識のほかに，図が直接記述していない知識も, 推論により導くことが可能となる. 例えば, Is_A の関係にある概念は, 上位の概念が持つ性質が下位の概念にも適用される．このような性質は，属性の継承（Inheritance）と呼ばれ，これを用いると直接関係が示されていない概念間の関係を導き出せる．図 2･1 の意味ネットワークでは，にわとりに羽があることは，直接示されているわけではないが，鳥が羽を持っているということから，にわとりに羽があることを推論することが可能となる．

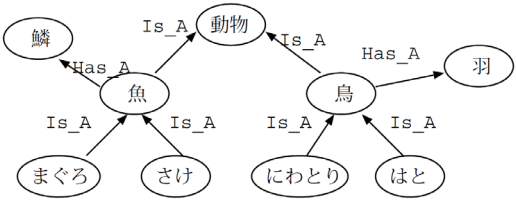

図 2・1　意味ネットワークの例

2-2-2 フレーム

フレーム（Frame）[2] は，Minsky によって提唱された知識を表現する枠組みである．フレームでは，ある概念に関連したものをフレームと呼ばれる一つのまとまりとして記述する．一つのフレームは，概念の名前を示すフレーム名，その概念が持つ属性を示すスロットより構成される．各スロットは，そのスロットがどういう属性を示しているのかを示すスロット名とスロ

ット名の中身となるファセットにより構成されている．**図 2･2** にフレームの例を提示する．この図の左上にある「鳥」がフレーム名を表しており，このフレームが鳥に関する概念をまとめたフレームであることを示している．その下にあるのがスロットを表している．この例の最初のスロットは，機能というスロット名に対して，「default 飛ぶ」というファセットが入っていることを示している．ファセットには，値を入れる value，デフォルト値を入れる default，プログラムの起動を促すデーモンの呼び出しが入る．value を指定したときは，値が直接入る．default を指定したときには，デフォルトとなる値が入る．デーモンの呼び出しが入っている場合には，その値が参照されたときに，計算のための手続きが呼び出され，値を計算するようになっている．例えば，太郎という特定の個人のフレームに，年齢のスロットがあった場合には，そのファセットは，参照する時期によって違う値が必要となる．そのため，生年月日に関するスロットなどを参照して，年齢を計算するプログラム（デーモン）が必要となる．

フレームでは，意味ネットワークと同様に，上位の概念からの属性の継承を利用したり，ほかのフレームに値を問い合わせしたりすることにより，推論を行う．

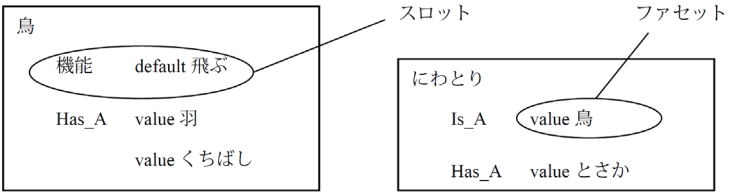

図 2・2　フレームの例

2-2-3 黒板モデル

黒板モデル（Blackboard Model）は，音声理解システム HEARSAY-II[1] で用いられた知識処理方式である．音声理解システムにおいては，音素レベルの処理に必要な知識，単語レベルの処理に必要な知識など，様々な種類の知識が必要となる．そのため，異なる種類の知識を統合して利用するための枠組みとして，黒板モデルが考えられた．黒板モデルでは，様々な知識を保持する知識ベースが分散して存在すると考える．また，それらの知識ベースを統合するための場として，黒板を考える．黒板には，各知識ベースが保持する知識を基に，推論した結果が書き込まれる．そして，その黒板の状況をほかの知識ベースが利用することによって，更に進んだ推論結果が黒板に書き込まれる．このような処理を繰り返すことによって，分散して存在する知識ベースが協調してあたかも一つの知識ベースのように推論を行うことが可能となる．

黒板モデルは，知識表現として用いられるのみならず，知識ベースを独立して構築できる利点があるため，分散環境に適応した知識システムアーキテクチャとしても用いられる．

2-2-4 プロダクションシステム

プロダクションシステム（Production System）は，Newell によって提案された知識処理方式である．プロダクションシステムは大きく分けて，3 つの要素から成り立っている．1 つ目は，プロダクション規則と呼ばれるもので，多くの規則を保持する．各プロダクション規則は，「if

条件 then 結論」の形式で表現され，様々な知識がこの形式によって記述されている．2 つ目は，作業記憶（Working Memory）と呼ばれる，データを短期的に記憶する場所である．ここに，現在のシステム外部の状態や作業中の状態が記録される．3 つ目は，これらの 2 つの要素を利用する推論部である．推論部は，全体の制御を司り，作業記憶に記録されている状態に基づき，規則の照合を行う．そして，その規則を適用することで，結論を得ることを繰り返す．しかし，実際には，複数のプロダクション規則が作業記憶の状態と一致することがあるため，そのなかからどの規則が選ばれるのかを制御する必要がある．その競合解消も推論部によって行われる．そして，結論が得られた後，その結論に基づいて，何かの行動を実行すると，作業記憶に格納されている状態が変化する．この変化した新しい状態に応じて，再びプロダクション規則を選択するということを繰り返し，システムの実行が進んでいく．

プロダクションシステムは，当初，人間の心のモデルとして研究されていた．その観点から，人間の記憶に対応付け，プロダクション規則の部分を長期記憶，作業記憶の部分を短期記憶と呼ぶこともある．

2-2-5 事例ベースシステム

事例ベースシステム（Case-based System）は，事例を知識として，たくさん蓄えることにより，問題の解決を図るシステムである．人間は，過去の事例を参考にして，しばしば問題の解決を行う．例えば，裁判においては，過去の判例（事例）を基に，類似の事件のときと同じような判決が出される．これと同じことを実現するのが，事例ベースシステムである．

事例ベースシステムでは，まず，過去の事例をデータベースに蓄える．データベースに格納する際には，事例の検索が行われやすいように，事例の特徴を表すような記述方法が使われる．例えば，医療診断の事例ベースシステムでは，患者の症状などの属性が使われるが，裁判の事例ベースシステムでは，当事者の関係を表現する意味ネットワークが使われるなど，その事例の特徴を表すのに適した表現が使われる．新しい事例に対処する際には，構築したデータベースから過去の事例が検索される．全く同じ事例があった場合には，過去と同じ対応をとることで問題の解決を行い，全く同じ事例が見つからない場合には，過去の類似した事例を修正することで，問題の解決を行う．そして，その結果に応じて，新しい事例をデータベースに登録することで，データベースを随時充実させていく．

事例ベースシステムは，医療診断や法律推論の分野などにおいて，しばしば用いられる手法である．

■参考文献

1) L.D. Erman, F. Hayes-Roth, V.R. Lesser, and D.R. Reddy : “The HEARSAY-II speech-understanding system: Integrating knowledge to resolve uncertainty,” Computing Surveys, 12(2), pp.213-253, 1980.
2) M. Minsky : “A framework for representing knowledge,” MIT AI Lab. Artificial Intellligence Memo No.306, 1974.

■S3 群－3 編－2 章

2-3 論理に基づく知識表現と推論

（執筆者：岩沼宏治）［2018 年 12 月 受領］

論理学の研究は，古くはアリストテレスの時代まで遡ると言われている．19 世紀には，それまでの自然言語の上に展開された論理学の曖昧さや不完全さを克服するために，記号に基づく論理学（数理論理学）が考案されている．ブールやド・モルガンにより創始された命題論理では，基本命題を 1 つの記号に単純化して表現し，複雑な命題（知識）は複数の命題記号と論理演算子を組み合わせた複合命題論理式で表現する．真偽値や充足可能性の計算（推論）は，どれも複合命題式の外形に基づいて行う．

フレーゲの述語論理は，より精密に命題（知識）を取り扱うために，命題を主語と述語に分離し，更に任意の対象を表す変数とその限量子（束縛子）の導入を行って記述力の本質的な拡張を図った論理体系である．述語論理の推論は，述語論理式の構造に基づく記号列計算のみによって行われる．

この 2 つの論理体系は，今日ではコンピュータサイエンス全般の重要な理論的な基盤となっている．数理論理は現実の知識や問題をより簡潔に表現し，高度な推論を実現するための基盤としても優れており，論理プログラミングや帰納推論など，従来の数理論理を拡張した体系が研究開発されている．

2-3-1 命題論理に基づく知識表現と推論

命題とは，真偽が定まる言明文である．命題論理では命題の詳細には立ち入らずに，命題を 1 つの記号に単純化・抽象化して表現するもので，現在の記号論理学のなかで最も単純な体系である．複雑な命題は，複数の命題記号を論理結合子で組み合わせて表現する．古くから論理関数や回路の設計問題の重要な理論基盤となっており，大規模集積回路の効率的な設計を実現するために，論理関数を圧縮表現する BDD 技術が 1980 年代の末に開発 [1] されている．人工知能の分野における実用知識や学習データは巨大な組合せ集合で表現されることが多く，その効率的な保持と高速な処理は極めて重要な問題となる．この問題に対して，BDD を改良した ZDD 圧縮技術が開発 [1] されている．ZDD は，現実の世界によく現れる疎な組合せ集合を非常にコンパクトに圧縮できる．また，組合せ集合の上の集合演算は，ZDD で圧縮した形のままで高速計算することができ，非常に有用である．

命題論理上の各種の推論問題は，その殆どが命題論理式の充足可能性判定問題（SAT：Satisfiability Problem）に帰着される．また，多くの組合せ（制約充足）問題は SAT 問題へ多項式時間還元できることから，SAT 問題を高速に解く SAT ソルバー（SAT Solver）は実用上とても重要 [2] である．1990 年代の末から，SAT ソルバーの計算実行過程での矛盾節の学習手法や決定命題変数の適応的な選択戦略，あるいは SAT ソルバーの再スタート戦略などの新しい技術が次々と開発され，SAT ソルバーの性能は飛躍的に向上している．近年では，数百万の命題変数からなる問題が実用時間内に解けるようになっている [3]．これにより，種々の組合せ問題を直接解かずに，一度 SAT 問題へ変換し，それを SAT ソルバーで解いて元問題の解を得る手法が非常に有効なものになってきている．述語論理の定理証明システムや論理プログラムの推論エンジンの核としても，SAT ソルバーが利用されることが多い．システム検証，プランニン

グ，スケジューリング，制約充足，制約最適化などの各種の問題を効果的に解くために，多種多様な SAT ソルバーが開発され，また SAT 問題への効果的な変換手法に関しても多数の研究が行われている．

2-3-2 一階述語論理に基づく知識表現と推論

一階述語論理では，命題を主語と述部に分離して基本命題の中身の詳細記述する．主語を記述するために変数，定数，関数を表す記号を導入し，述部の表現に述語記号を導入する．変数記号は基本的に不特定な対象を表すが，表現能力を向上させるために「すべての対象 x に対して... 」を表す全称限量子$\forall x$や「ある対象 x が存在して... 」を表す存在限量子$\exists x$が導入される．この変数と限量子の導入により，一階述語論理は無限個の対象に関する性質を記述できる能力を持つ．その記述能力は命題論理よりも本質的に高いが，そのために逆に，自動推論や真偽値の完全な計算はかなり困難になる．無限個の対象を扱うような直接的な真偽値計算は本質的に不可能であるので，一階論理上の推論は，論理式（記号列）の構造計算に基づく恒真（Valid）な論理式の証明（導出）計算が基本となる．恒真な論理式は，記号の意味解釈によらず真となる論理式である．そのため，論理式が恒真であるか否かは，論理式を構成している記号列の形だけで定まることに注意して欲しい．

一階論理に基づく知識表現と推論システムの最も代表的なものとしては，1980 年代から多くの研究がなされた論理プログラミング（Logic Programming）[4),5)] が挙げられる．プログラム言語 Prolog は述語論理の表現能力の高さを活かしつつ，自動推論を容易にした最も基本的な論理プログラミングの実装例である．Prolog のプログラム Σ は，確定節（Ddefinite Clause）と呼ばれる $A \leftarrow B_1, B_2, \cdots, B_m$ なる形をした式の集合である．ここで，$A, B_1, B_2, \cdots, B_m\ (m \geq 0)$ は変数 $x_1, \cdots, x_k\ (k > 1)$ を含んだ原子論理式である．確定節の論理的な意味は $\forall x_1 \cdots x_k\ (B_1 \wedge B_2 \wedge \cdots \wedge B_m \supset A)$ なる述語論理式と同じである．

プログラム論的には，A はプログラムの頭部（名前や引数の宣言部）を表している．$B_1, B_2, \cdots, B_m$ はプログラムの実行文の列であり，ボディと呼ばれる．A が真であることを示すためには，ボディ部の $B_1, \cdots, B_m$ がすべてが真であることを示す必要があるという計算（証明）手続きを表現している．各 B_i の計算には，B_i を頭部に持つ確定節を用いて，そのボディ部を計算することになる．この後ろ向き推論を再帰的に繰り返し，$m = 0$ の確定節にたどり着くまで続ける．$m = 0$ のときの確定節 $A \leftarrow$ は単位節（Unit Clause）もしくはファクト（Fact）と呼ばれ，A が無条件に真であることを表している．実際には A や B_i には変数が含まれているので，単一化（Unification）と呼ばれる双方向の変数への代入操作により，必要な原子論理式の同一化を行って後ろ向きの再帰計算を進めていく．この計算法は SLD 導出（SLD-resolution）と呼ばれ，最初の計算目標（ゴール（Goal）と呼ばれる）A から始まるトップダウン型の計算（証明）手法である．

プログラム Σ の宣言的意味は，Σ のエルブランモデルの共通部分である最小モデルで与えられる．手続的意味は Σ 上の SLD 導出などの証明手続きなどで与えられる．この両者の意味が本質的に等価であることが証明できる [4)] ことから，論理プログラムは知識の宣言的記述とその計算を明確に分離できる枠組みとなっている．

人工知能における知識表現言語では，不完全で不確定な情報下での推論能力が求められる．論理プログラミングでの不完全情報の取扱いを可能にする計算原理に，失敗による否定（NAF:

Negation as Failure）がある．これは，プログラムΣ上での SLD 導出がすべて有限失敗（Finite Failure）する原子論理式 A を偽と判定するものである．これを用いれば，「A が偽である」ことをプログラム（知識）中に陽に記述しなくとも，推論計算の有限失敗により推定することが可能になる．NAF は閉世界仮説（Closed World Assumption）に基づく非単調な常識推論を一部実現しており，人工知能研究の最大の難問の一つであるフレーム問題（Frame Problem）[6] をも部分的に解決する非常に重要な計算原理である．NAF は Prolog 言語の開発研究のなかで提案され，高速な処理系も同時に開発されているが，その論理的な意味は通常の述語論理の否定とは異なるため，新しい意味論が必要となる．これに対しては述語の完備化（Predicate Completion）という手法に基づく明快な宣言的意味論が構築 [4] されており，ある条件下で，この完備化に基づく意味論は有限失敗に基づく手続的意味論と一致することが明らかになっている．

不確定な情報を取り扱うものとしては選言論理プログラミング（Disjunctive Logic Programming）が最も基本的である．そのプログラムは，選言節（Disjunctive Clause）と呼ばれる A_1; … ; A_n ← B_1, …, B_m なる形をした論理式の集合である．選言節は述語論理的な観点からは $\forall x_1 \cdots x_k \ (B_1 \wedge \cdots \wedge B_m \supset A_1 \vee \cdots \vee A_n)$ なる論理式と同等である．選言論理プログラミングはその性質上，プログラムの極小モデルは複数存在することが通常である．その解もすべての極小モデルで成り立つものと，どれか一つで成り立つものなどの区別が必要となり，推論計算はそれらを考慮して行う必要があることから，Prolog よりもかなり複雑になる．

近年では，不完全情報と不確定情報，及び論理的否定と失敗による否定のすべてを同時に取り扱うことができる枠組みが盛んに研究されており，最も有力な枠組みは解集合プログラミング（Answer Set Programming）[5] と呼ばれている．最も一般的な解集合プログラムは以下の一般節から構成されている．

$$L_1; \cdots ; L_k ; not\ L_{k+1}; \cdots ; not\ L_l \leftarrow L_{l+1}, \cdots , L_m, not\ L_{m+1}, \cdots, not\ L_n \qquad (4 \cdot 1)$$

ただし，$n \geq m \geq k \geq 0$である．*not* は NAF 演算子であり，各 L_i は原子論理式 A あるいはその否定 $\neg A$ である．演算子¬は論理的な否定を表している．解集合プログラミングは制約プログラミングの概念と関連が深く，その表現能力は非常に高い．解集合プログラミングの枠内での帰納推論（学習）も表現できることが知られている．よって当然のことながら，その一般的な推論計算の効率的な実現は非常に難しい．このため幾つかの部分体系，例えば，式(4･1)の頭部に *not* を含まない（すなわち，k=1）ような部分体系（拡張選言プログラム（Extended Disjunctive Program）と呼ばれる）などに対して処理系が開発実装されている．これまでに開発された処理系は，そのすべてがボトムアップ型計算を行うものとなっている点は大きな特徴である．

■参考文献

1) 湊　真一：“BDD/ZDD を基盤とする離散構造と演算処理系の最近の展開，”IEICE Fundamentals Review, vol.4, no.3, pp.224-230, 2011.
2) 井上克巳，田村直之：“SAT ソルバーの基礎，”人工知能学会誌, vol.25, no.1, pp.57-67, 2010.
3) 鍋島英知, 宋　剛秀：“高速 SAT ソルバーの原理，”人工知能学会誌, vol.25, no.1, pp.68-76, 2010.
4) J.W. ロイド(著)，佐藤雅彦，森下真一(訳)：“論理プログラミングの基礎，”産業図書, 1987.
5) 井上克巳，坂間千秋：“論理プログラミングから解集合プログラミングへ，”コンピュータソフトウェア, vol.25, no.3, pp.20-32, 2008.
6) J. McCarthy and P.J. Hayes：“Some philosophical problems from the standpoint of artificial intelligence，”Machine Intelligence, vol.4, pp.463-502, 1969.

■S3 群－3 編－2 章

2-4 制約に基づく推論

（執筆者：平山勝敏）［2008 年 11 月 受領］

制約という形式で宣言的に表現された知識を用いて推論を行う際，効率が一つの大きな課題となる．本節では，制約を効率良く処理するための基盤技術である制約充足問題とその解法に関する研究を紹介する．

2-4-1 制約充足問題

制約充足問題（Constraint Satisfaction Problem）は，変数集合 $X = \{x_1, \ldots, x_n\}$，値域集合 $D = \{D_1, \ldots, D_n\}$，制約集合 $C = \{C_1, \ldots, C_m\}$ からなり，一般にそれらの三つ組 (X, D, C) で表現される．任意の変数 x_i は，それに対応する値域 D_i より値 d_i をとるが，とることができる値は制約集合 C により制限されている．制約 C_j とは，変数のある部分集合 S_j 上で定義された関係（Relation）であり，一般に，S_j に含まれる変数群が同時にとることができる値の組合せを表す．制約充足問題 (X, D, C) の解とは，制約集合 C に含まれるすべての制約を充足するすべての変数への値の割当てである．

制約充足問題の例にグラフ彩色問題（Graph Coloring Problem）がある．これは，与えられた無向グラフに対し，辺で結ばれた任意の 2 頂点が異なる色となるように，グラフの頂点全体を指定された色を用いて塗り分ける問題である．また，近年，多くの実用的な問題に応用され注目されている充足可能性判定問題（Satisfiability Problem）も制約充足問題の一種である．これ以外にパズル問題のいくつか（例えば，n クイーン問題，クロスワードパズル，数独など）も制約充足問題とみなすことができる．

制約充足問題は NP 完全問題であり，あらゆる問題例の解を効率的に求めるアルゴリズムは事実上存在しない．しかしながら，1970 年代より主に人工知能の分野で精力的な研究が行われ，これまでに多くの成果が蓄積された．以下では，制約充足問題の代表的な解法を概観する．

2-4-2 制約充足問題の解法

制約充足問題の解法は大きく 2 つに分類できる．一つは推論（Inference）による解法，もう一つは探索（Search）による解法と呼ばれる．

(1) 推論による解法

推論による解法では，局所的に無矛盾な変数への値の割当てが存在するように問題例自体を変形することを目標とする．したがって，推論による解法では解を求めることは一般にはできない．しかし，そのような変形が多項式時間で行えるというメリットがある．通常，推論による解法は，後述する探索による解法の前処理として用いられるか，あるいは，探索による解法の部分手続きとして用いられる．

推論による解法では，どの程度まで局所的な無矛盾性を達成するか指定する必要がある．最もよく利用されるのは辺整合（Arc Consistency）という概念である．これは，任意の変数の値域内の任意の値について，ほかの各変数の値域内にそれと矛盾しない値が 1 つ以上存在するという状態を表す．この辺整合という状態を実現する辺整合アルゴリズムには AC-3 [6] や AC-4 [8] などがある．AC-4 は理論上最小コストで辺整合を実現するアルゴリズムだが，実際の性能は

AC-3 のほうが良いと言われている．なお，辺整合アルゴリズムは AC-4 以降も更に改良が進められている．また，辺整合の概念を拡張した路整合（Path Consistency）[6]，更に，これらを一般化した *k* 整合（*k*-consistency）[3] という概念も提案されている．

(2) 探索による解法

一方，探索による解法では，解を求めることを目標とする．最も基本的な解法は後戻り探索（Backtracking Search）と呼ばれる深さ優先探索で，その概要は，

- Look-ahead 操作：変数を 1 つ選択し，現在の部分解と制約違反を起こさない値をそれに代入して部分解を伸ばす．
- Look-back 操作：選択された変数に対して，値域内のどの値を代入しても現在の部分解との制約違反が避けられないとき（すなわち Look-ahead 操作が行き詰まったとき），現在の部分解内のある変数に戻り，その値を変更する．

という 2 つの基本操作を繰り返すというもので，通常，解を 1 つ以上発見するか，あるいは，解がないことを発見して終了する．しかし，後戻り探索を工夫なく単純に実装すると，一般に探索効率が非常に悪い．そこで探索効率を上げるための様々な工夫が提案されている．以下では，後戻り探索を拡張する Look-ahead 戦略と Look-back 戦略，及び，後戻り探索とは全く異なるアイデアに基づく確率的局所探索を紹介する．

(a) Look-ahead 戦略

Look-ahead 操作の際，どの変数を選択するか，また，その変数に対してどの値を選択するかは将来の探索効率に大きな影響を与える．Look-ahead 戦略では，操作に先立って前述の推論アルゴリズムを実行し，その結果，最も効果的と思われる変数や値を選択する．なお，どの程度の推論を行うべきかは問題に依存する．従来，フォワードチェッキング（Forward Checking）という限定的な弱い推論を行うのが良いとされていたが [5]，近年，規模が非常に大きい困難な問題に対しては，前述の辺整合アルゴリズムや更に強い推論を行うほうが良いと報告されている．

(b) Look-back 戦略

一方，Look-back 操作の際には，現在の部分解内のどの変数に戻るかが重要となる．この選択を誤ると，部分解内の本質的な矛盾が解消されずに同じ行き詰まりに何度も遭遇するスラッシング（Thrashing）という現象が起こり，探索効率が著しく低下する．これを避けるには，部分解内の矛盾の原因となる変数を特定し，その変数に直接ジャンプするバックジャンピング（Backjumping）[9] が有効である．また，後戻りの際，部分解内の矛盾の原因を解析し，それを制約として記録することにより以降の探索において類似した行き詰まりに陥らないようにする Nogood 学習（Nogood Learning）[2] も非常に有効である．

(c) 確率的局所探索

上のように工夫したとしても，後戻り探索の最悪時のコストは一般に変数の数の指数オーダとなる．そのため，「近似解」を効率的に求める確率的局所探索（Stochastic Local Search）が提案されている [7]．基本的な考え方は，まず，すべての変数への値の初期割当てを適当な方法で求め，以後，割当てに対し局所的な修正（Local Repair）を繰り返し行いながら制約違反を次第に解消していくというものである．この手法は，局所的な修正では制約違反を解消できない局所解に陥る可能性があり，そのための対策が不可欠である．その対策として，ランダムな初期

割当てから再スタートしたり，あるいは，局所的な修正の代わりにランダムな割当て変更をある確率で行うなど，確率的な振る舞いを導入することが有効である．なお，確率的局所探索は，解が存在する問題に対して，多くの場合にその解を発見することができるが，解が存在しない問題に対して，解が存在しないという事実を発見することができない．

2-4-3 その他の話題

以上，制約充足問題とその代表的な解法を概観した．最後にその他の話題をいくつか挙げて結びとする．

1990 年代前半に，グラフ彩色問題，充足可能性判定問題など代表的な制約充足問題において，問題の構造に関係するある特定のパラメータ空間上で，ランダムに生成した問題例の平均的な複雑さが急激に変化する相転移（Phase Transition）と呼ばれる現象が報告された [1]．その後，多くの研究者により，制約充足問題の「真の複雑さ」に関するより詳細な理論的及び実験的な解析が進められた．

また，現実問題への応用を想定し，制約充足問題自体を拡張する試みもなされている．特に，制約をコスト関数に置き換えて，コストの総和を最小化する変数への値の割当てを求める制約最適化問題（Constraint Optimization Problem），及び，制約充足問題の構成要素が複数のエージェントに分散された分散制約充足問題（Distributed Constraint Satisfaction Problem）は，その提案以来 [4),10)]，現在に至るまで継続的に研究されている．

■参考文献

1) P. Cheeseman, B. Kanefsky, and W.M. Taylor：“Where the really hard problems are,”Proceedings of the 12th International Joint Conference on Artificial Intelligence, pp.331-337, 1991.
2) R. Dechter：“Enhancement schemes for constraint processing: backjumping, learning and cutset decomposition,”Artif. Intell., vol.41, pp.273-312, 1990.
3) E.C. Freuder：“Synthesizing constraint expressions,”Commun. ACM, vol.21, no.11, pp.958-966, 1978.
4) E.C. Freuder and R.J. Wallace：“Partial constraint satisfaction,”Artif. Intell., vol.58, no.1-3, pp.21-70, 1992.
5) R.M. Haralick and G.L. Elliot：“Increasing tree-search efficiency for constraint satisfaction problems,”Artif. Intell., vol.14, pp.263-313, 1980.
6) A.K. Mackworth：“Consistency in networks of ralations,”Artif. Intell., vol.8, no.1, pp.99-118, 1977.
7) S. Minton, M.D. Johnston, A.B. Philips, and P. Laird：“Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems,”Artif. Intell., vol.58, no.1-3, pp.161-205, 1992.
8) R. Mohr and T.C. Henderson：“Arc and path consistency revisited,”Artif. Intell., vol.28, pp.225-233, 1986.
9) P. Prosser：“Hybrid algorithms for constraint satisfaction problems,”Computational Intelligence, vol.9, no.3, pp.268-299, 1993.
10) M. Yokoo, E.H. Durfee, T. Ishida, and K. Kuwabara：“The distributed constraint satisfaction problem: formalization and algorithms,”IEEE Trans. Knowledge and Data Engineering, vol.10, no.5, pp.673-685, 1998.

■S3 群－3 編－2 章

2-5 不確実性を扱う知識表現と推論

（執筆者：亀谷由隆）［2016 年 7 月 受領］

現実世界を対象としたとき，そこで何が起こるのかを網羅的に記述したり，実際に何が起こっている/いたのかを漏れなく観測したりするのは困難である．したがって，このような知識や観測の不完全性に由来する不確実性を扱うための方法が必要となり，これまでの人工知能研究において様々な手法が提案されている．まず，初期の研究においては，医療用エキスパートシステム MYCIN の確信度（Certainty Factor：CF），Dempster-Shafer 理論，Nilsson の確率論理（Probabilistic Logic）などが知られている [4]．また，ファジィ論理（Fuzzy Logic）も同様の目的を持つ．更に，論理に基づく知識表現・推論におけるデフォルト論理などの枠組みも上記のような知識や観測の不完全性に対処するためのものと考えることができる．

例えば，MYCIN では以下のように IF-THEN 規則を記述する [2]．

IF

E_1：The stain of the organism is gramneg（染色すると有機体はグラム陰性を示した）**AND**

E_2：The morphology of the organism is coccus（有機体は球状である）**AND**

E_3：The growth conformation of the organism is chains（有機体は鎖状に伸びている）

THEN

H：There is suggestive evidence (0.7) that the identity of the organism is strepococcus

（その有機体は連鎖球菌であるという証拠 (0.7) がある）

後件部に現れる 0.7 はこの規則 R の確信度 CF(R)である．前件部の各条件の確信度が CF(E_1) = 0.5，CF(E_2) = 0.6，CF(E_3) = 0.3 であったとすると，前件部全体の確信度は CF($E_1 \wedge E_2 \wedge E_3$) = min{CF($E_1$), CF($E_2$), CF($E_3$)} = 0.3 となる．最終的に結論 H の確信度は CF($E_1 \wedge E_2 \wedge E_3$)･CF($R$) = 0.21 と求まる．

近年の人工知能研究では，確率の概念を使って不確実性を扱うアプローチがよく用いられる．この背景としてグラフィカルモデル（Graphical Model）研究の発展があったことが挙げられる．グラフィカルモデルはグラフ構造で表現される確率モデルであり，(i) 確率論に基づく明確な意味論を備える，(ii) 事象間の依存性・独立性をグラフ構造で自然に表現できる，(iii) 事象間の独立性に基づく効率的な厳密・近似推論アルゴリズムが利用可能である，などの特徴を持つ．

これらの特徴に関して，まず (i) より推論結果の妥当性に根拠を与えることができる．これは上で述べた MYCIN の確信度の計算規則に欠けていた点である．そして (ii) は，知識表現形式としても優れていることを示しており，(i) と合わせて他分野の専門家との共同作業にも有効に働く．更に (iii) は，（計算機の高速化と相まって）初期の人工知能研究における壁であった確率計算に要する計算量を克服するのに重要な役割を果たした．また，データからモデルの構造・パラメータを学習して統計的な推測を行うことも可能であり，グラフィカルモデルは人工知能分野における不確実性を考慮した知識表現と統計的な機械学習を自然に結び付ける存在となっている．本節ではこれ以降，代表的なグラフィカルモデルであるベイジアンネット（Bayesian Network）[1),3),7)] とその周辺技術を簡単に紹介する．

ベイジアンネットでは，興味ある不確実な現象を確率的に表現するために，まず n 個の確率変数の集合 $\boldsymbol{X} = \{X_1, X_2, \cdots, X_n\}$ を導入し，その上の同時分布 $P(X_1, X_2, \cdots, X_n)$ を考える．簡単のため各 X_i は離散確率変数とする．そして，$X_1, X_2, \cdots, X_n$ の各々に対応する節点からなる有向非循環グラフ（Directed Acyclic Graph，以降 DAG）によって同時分布における条件付き独立性（Conditional Independence）が表現される．人間の直観に合うように，DAG 中の有向辺の向きを因果の向きと捉えて考えることも多い．ここで，DAG における各 X_i の親節点の集合を $\mathbf{Pa}_i$ と書き，X_i の子孫でない節点の集合を $\mathbf{Nd}_i$ と書く．$\mathbf{Pa}_i$ 内の変数に値が与えられたとき X_i と $\mathbf{Nd}_i$ が条件付き独立になる（マルコフ条件）と仮定すると，同時分布 $P(X_1, X_2, \cdots, X_n)$ は $\prod_{i=1}^{n} P(X_i|\mathbf{Pa}_i)$ と各 X_i に関する条件付き確率の積に分解される．本来大域的な情報である同時分布を局所的な条件付き確率の積に分解したことで確率推論・学習アルゴリズムが大幅に高速化される．

ベイジアンネットの確率推論においては，まず証拠変数 $\boldsymbol{E}$ と質問変数 $\boldsymbol{Q}$ を考える（各々複数の変数を含んでよい）．我々は $\boldsymbol{E}$ の値 $\boldsymbol{e}$ を観測し，これを証拠（Evidence）と呼ぶ．一方，$\boldsymbol{Q}$ が確率推論の対象となる．このときベイジアンネットでは，

- 証拠の生起確率 $P(\boldsymbol{e})$，周辺確率 $P(\boldsymbol{q})$，条件付き確率 $P(\boldsymbol{q}|\boldsymbol{e})$ の計算
- MPE（Most Probable Explanation）推論：$\boldsymbol{Q} = \boldsymbol{X} \setminus \boldsymbol{E}$ に対する $\mathrm{argmax}_q P(\boldsymbol{q}|\boldsymbol{e})$ の計算
- MAP（Maximum A Posteriori）推論：$\boldsymbol{Q} \subset \boldsymbol{X} \setminus \boldsymbol{E}$ に対する $\mathrm{argmax}_q \Sigma_u P(\boldsymbol{q},\boldsymbol{u}|\boldsymbol{e})$ の計算（$\boldsymbol{U} = \boldsymbol{X} \setminus (\boldsymbol{Q} \cup \boldsymbol{E})$）

といった確率推論が行われる（慣例に従って確率変数は大文字で表記し，その値は対応する小文字で表記する）．一般にこれらの確率推論は NP 困難であることが知られているが，実問題のネットワークの多くに対して現実時間で動作する厳密推論手法として，変数消去法（Variable Elimination），接合木法（Junction Tree, Join Tree）などが知られている．また，近似推論としては MCMC（Markov chain Monte Carlo）法をはじめとするサンプリング手法，平均場近似手法などが用いられる．更に，無向ループを持たない（Singly Connected）DAG を前提とする J. Pearl の信念伝播法（Belief Propagation：BP）を無向ループを持つ（Multiply Connected）DAG に適用する Loopy-BP という近似推論手法も 1990 年代終わりに提案された．

データからベイジアンネットのパラメータ（の事後分布）や DAG 構造そのものを学習する試みも広く行われている．パラメータ学習では最尤推定，MAP（Maximum a Posteriori）推定，ベイズ推定，あるいは不完全データに対する EM（Expectation-maximization）アルゴリズムの利用など，従来の統計的推測手法を標準的なやり方で適用できる．また，構造学習に対しては大きく分けて制約に基づく手法とベイズスコアに基づく手法の 2 つがある．前者は統計的検定による条件付き独立性の検査と有向辺の向きに関する制約を利用してベイジアンネットの構造を決定する．代表例として Spirtes らの PC アルゴリズムが知られる．一方後者では，ベイズ的立場に基づき，所与のデータ D の周辺尤度（Marginal Likelihood）$P(D|G) = \int P(D|G,\theta)P(\theta|G)\,d\theta$ を最大にする DAG 構造 G を求める方法が典型的である．周辺尤度の計算はパラメータ θ の積分を伴い，厳密計算は困難であるため，MCMC 法によるサンプリング近似や背後の確率分布に仮定を入れるなどして導出した近似スコア（BIC（Bayesian Information Criterion）が代表例）が用いられる．また，可能な DAG 構造の数は膨大であることから，局所探索，ビーム探索，分枝限定法といった探索技法が応用されている．

ベイジアンネットを含むグラフィカルモデルは 1980 年代終わりから 2000 年代にかけて理論

的整備が進んだが，ほぼ同時並行でグラフィカルモデルを拡張した知識表現・機械学習技術の研究も進められている．まず，グラフィカルモデルの拡張記法として，同種の節点の繰り返しをプレートで表記したプレート記法が広く使われている．これを発展させ，事象間の依存関係をより記述力の高い知識表現言語で記述し，そこからグラフィカルモデルを直接/間接的に構築して実質的な推論や学習を行う手法（一般に Knowledge-based Model Construction（KBMC）と呼ばれる）が 1990 年代終わりから 2000 年代前半にかけて数多く提案され，統計的関係学習（Statistical Relational Learning：SRL）という分野を成している [6]．なかでも，関係データベース中の非キー値の確率的な依存関係をベイジアンネットで表現する Koller らの PRM（Probabilistic Relational Model），一階述語式に基づきマルコフネット（Markov Network，無向グラフに基づくグラフィカルモデル）を規定する Domingos らの MLN（Markov Logic Network）が代表的である．

一方，論理プログラミング分野においては，最小モデル意味論の確率的拡張として分布意味論（Distribution Semantics）が提案されており，現在，統計的関係学習と近い立場にある帰納論理プログラミング（Inductive Logic Programming：ILP）分野では分布意味論が一つの標準的な確率的意味論となっている．また，確率モデル構築におけるプログラミング的側面が強調された確率プログラミング（Probabilistic Programming）が 2000 年代後半から研究されており，前述の分布意味論に基づき論理型言語 Prolog を拡張した PRISM は初期の試みの一つとみなせる．同分野では他にも Pfeffer らの IBAL（その後継 Figaro），Goodman らの Church が関数型言語を基に開発されている．更に，関係データベースにおける不確実性を扱う手法として，確率的データベース（Probabilistic Database）[5] と呼ばれる枠組みもデータ工学分野において独自に発展している．

■参考文献

1) A. Darwiche：“Modeling and Reasoning with Bayesian Networks,”Cambridge University Press, 2009.
2) J.C. Giarratano and G.D. Riley：“Expert Systems: Principles and Programming,”Course Technology, 2005.
3) D. Koller and N. Friedman：“Probabilistic Graphical Models,”The MIT Press, 2009.
4) S. Russell and P. Norvig：“エージェントアプローチ人工知能（第 2 版）,”共立出版, 2003.
5) D. Suciu, D. Olteanu, C. Rè, and C. Koch：“Probabilistic Databases,”Morgan & Claypool Publishers, 2011.
6) 佐藤泰介，亀谷由隆：“グラフィカルモデルにおける論理的アプローチ,”人工知能学会誌, vol.22, no.3, pp.306-319, 2007.
7) 植野真臣：“ベイジアンネットワーク,”コロナ社, 2013.

■S3 群－3 編－2 章

2-6 生命にならう知識表現と推論

（執筆者：山田武士）［2008 年 10 月 受領］

生命にならう知識表現と推論の代表的なアプローチとして，進化的計算（Evolutionary Computation : EC）が挙げられる．進化的計算とは，自然界の生命の進化のプロセスに多少なりとも基づいている計算手法の総称であり，進化的アルゴリズム（Evolutionary Algorithms）や，群知能（Swarm Intelligence）などの一連の手法から成る．

進化的アルゴリズムは，主にダーウィン（Charles Robert Darwin）の進化論の考え方に基づき，自然界の進化と淘汰のメカニズムを取り入れた最適化アルゴリズムである．進化的アルゴリズムは，組合せ最適化問題など，通常の方法では扱えないような，高度に複雑な目的関数の最適化に適しており，これまで，機械学習や，巡回セールスマン問題，スケジューリング問題，グラフ分割問題などの組合せ最適化問題をはじめ，様々な問題に適用され成果を挙げている．例えば，新幹線の「N 700 系」車両の先頭形状の最適化にも用いられた．

進化的アルゴリズムの代表例は，遺伝的アルゴリズム（Genetic Algorithms：GAs）である．遺伝的アルゴリズムは 1975 年にミシガン大学のジョン・ホランド（John H. Holland）が提案し，その基礎となる一連のスキーマ定理を発表した後，当時ホランドの学生であった，デイビッド・ゴールドバーグ（David E. Goldberg）が，ガスパイプライン制御への適用に成功したのを契機に注目を集めるようになった．彼は 1989 年に書籍 [1] を出版し，遺伝的アルゴリズムを更に広めた．

一方，1966 年にローレンス・フォーゲル（Lawrence Fogel）によって提案された進化的プログラミング（Evolutionary Programming）や，1960 年代にレッケンベルグ（Ingo Rechenberg）やシュウェーフェル（Hans-Paul Schwefel）らによって提唱された進化的戦略（Evolution Strategies）なども，同様の考え方に基づく計算モデルである．これらは当初，独立に提案されたが，今日では，進化的アルゴリズムという名のもと，統一的に扱われている．これらの手法は主に，対象となる問題の解を進化させる点で共通しているが，1990 年代の初めにジョン・コーザ（John R. Koza）は，コンピュータプログラム自体を進化させる手法である遺伝的プログラミング（Genetic Programming）を提案した．また，与えられた入力に対する最適なアクションを実行するルールを適応的に学習する，学習分類システム（Learning Classifier System）も進化的アルゴリズムの一つである．これらの詳細については，文献 2）を参照されたい．

一方，群知能は，蟻が集団で行動し，フェロモンの分泌を通じてお互いに情報交換しながら最短経路を探索する様子をモデル化した蟻コロニー最適化（Ant Colony Optimization），及び鳥や魚の群れに見られる社会的行動にヒントを得た，集団ベースの確率的最適化手法である，粒子群最適化（Particle Swarm Optimization）などから成る [3]．

また，進化的計算は，物理における焼きなましのプロセスに習ったシミュレーテッド・アニーリング（Simulated Annealing）や，タブーリストと呼ぶ効率的なメモリ構造を駆使するタブー探索法（Tabu Search）などとともに，メタ戦略（Metaheuristics）と呼ばれている．

進化的アルゴリズムにおける，集団による確率的最適化の考え方は，マルコフ連鎖モンテカルロ（Markov Chain Monte Carlo：MCMC）法の並列化である，集団モンテカルロ法の一形態，と考えることもできる．ただし，集団モンテカルロ法を含む，MCMC 法では確率分布の推定，

もしくは，期待値の計算が主眼であるのに対し，進化的アルゴリズムはあくまでも目的関数の最適値の探索に重点が置かれている．最近は，前者の要素を取り入れ，探索空間全体に対し，有望な解候補の確率分布を直接推定する，分布推定アルゴリズム（Estimation of Distribution Algorithm：EDA）や，確率分布のモデルとしてベイジアンネットワーク（Bayesian Network）を用いる，ベイジアン最適化アルゴリズム（Bayesian Optimization Algorithm：BOA）も提案されている[4]．**図 6･1** にメタ戦略における各種手法の位置づけを示す．本節では進化的アルゴリズムに焦点を当て，そのなかでも，主に遺伝的アルゴリズムを取り上げ，その基本的枠組みについて説明する．

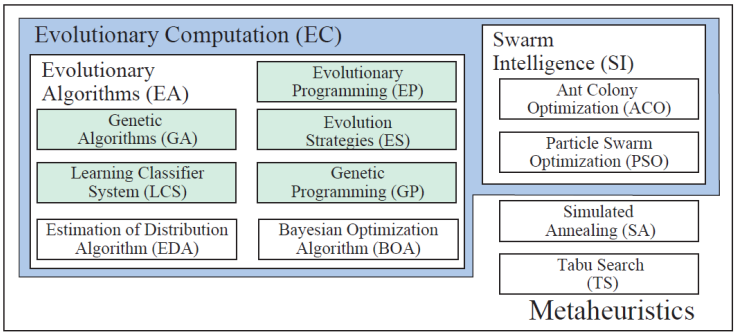

図 6・1　各種手法の位置づけ

2-6-1 進化的アルゴリズムにおける基本概念

進化的アルゴリズムで扱われる基本的概念には，遺伝学の影響を受けた用語が多く用いられている．まず，最適化すべき問題の解の候補を，個体（Individual）と呼ぶ．一般に進化的計算では，複数の個体からなる集合を扱い，これを個体集団（Population）と呼ぶ．ここで，各個体は，表現型（Phenotype）と遺伝子型（Genotype）と呼ばれる 2 つの表現を持つ．表現型は最適化すべき問題の解の候補そのものを表し，最適化すべき問題で通常用いられる解表現をそのまま用いることができる．一方，遺伝子型は，染色体（Chromosome）もしくはゲノム（Genome）という形に符号化された解表現を表す．染色体は遺伝子（Gene）を通常，直線状に並べたもので表現され，それぞれの遺伝子は複数の値，すなわち，対立遺伝子（Allele）を取りうる．これらの値によって，対応する形質や特徴を継承するかどうかを制御する．

例えば，染色体が 0, 1 から成るビット列で表現されている場合，特定の場所（Locus）の遺伝子の値が 1 であれば対応する形質がオン（すなわち継承する）を意味し，0 であればオフ（継承しない）を意味する．対象となる最適化問題に応じて，実数列，記号の順列や行列など，より複雑な表現を用いる場合もある．

個体集団内の個々の個体は部分的な変更を受け，新たな個体を生成する．生成された新たな個体を子孫（Offspring）もしくは単に子（Child）と呼ぶ．現在の個体集団がその子孫によって置き換えられ，新しい個体集団が生成されるとき，この新しい個体集団を新たな世代（Generation）と呼ぶ．更に，これら全体の最適化のプロセスを進化（Evolution）と呼ぶ．

2-6-2 個体の適応度

各個体は，自身を取り巻く環境にどれくらい適合しているかの尺度である，適応度（Fitness）を持つ．ダーウィンの進化論によると，個体集団内の個体のうち，環境に最も適したものが次の世代に最もたくさんの個体を残す．これは別名「適者生存」（Survival Of The Fittest）と呼ばれる．

進化的アルゴリズムを用いて最適化問題を解く場合，その問題が最適化すべき目的関数が，いわば環境の役割を果たす．すなわち，個体の適応度は，その個体を対応する最適化問題の解候補と見たとき，最適化の基準に照らし合わせてどれくらい「良い」か，に対応すると考えられる．したがって，解くべき最適化問題が最大化問題であれば，適応度は目的関数値を直接用いることもできるし，何らかのスケーリングを行う場合もある．

最適化問題が最小化問題の場合は，現個体集団内における各個体が持つ目的関数の最大値から自分自身の目的関数値を引き算するなど，変換を施す．もしくは，ランキングを用いることもできる．すなわち，現個体集団の各個体をその目的関数の値が悪いものから良いものへとソートし，その順位を適応度とする．

2-6-3 単純な遺伝的アルゴリズム

まず，遺伝子型が長さ n のビット列で表現されるような，最も単純なケースを考える．ビット列を対象とする単純な遺伝的アルゴリズム（Simple Genetic Algorithm）は，交叉（Crossover），突然変異（Mutation），生殖（Reproduction）の 3 つの遺伝的オペレータで構成される．交叉と突然変異は合わせて遺伝的組み換え操作（Genetic Recombination Operators）と呼ばれる．

(1) 交　叉

個体集団内の各個体をランダムに 2 個体ずつペアとし，これを両親（Parents）と呼ぶ．交叉はこの両親と呼ばれる個体ペアの遺伝子型に対して作用し，両親の形質を部分的に受け継いだ，新たな個体（通常は 2 個体）を生成する操作である．生成された個体は，子，もしくは，子孫と呼ばれる．基本的には，両親の 2 個体の染色体（ゲノム）のそれぞれをいくつかの断片に分解し，それを再び組み合わせることによって新たな染色体を生成する．一点交叉（1-point Crossover），二点交叉（Two-point Crossover），一様交叉（Uniform Crossover）などがよく用いられる．例として，二点交叉では，2 点の交叉点をランダムに設定し，その内側を一つの親から，外側をもう一つの親から受け継いで，新たなビット列を生成する．

$$\begin{array}{lcl} P_1 = 00\,|\,0000\,|\,0000 \\ P_2 = 11\,|\,1111\,|\,1111 \end{array} \Longrightarrow \begin{array}{lcl} K_1 = 00\,|\,1111\,|\,0000 \\ K_2 = 11\,|\,0000\,|\,1111 \end{array} \qquad (6\cdot 1)$$

(2) 突然変異

突然変異は単一の個体へ作用し，いわば，染色体複製の際に生じるエラーに相当する．基本的なビット反転突然変異を以下に示す．

$$P = 0000000000 \Longrightarrow K = 0000100000 \qquad (6\cdot 2)$$

(3) 再　生

再生もしくは生殖（Reproduction）とは，個体集団内の各個体が，それぞれの適応度（Fitness）に応じた個数の複製を次の世代に残すことを意味する．通常，各個体は，適応度に比例する個

数の個体を残す．したがって，適応度の高い個体ほど，多くの複製を残すことになる．すなわちこれは，人工的な自然淘汰（Natural Selection）と言える．

最も単純な再生操作は，ルーレット選択（Roulette Wheel Selection）と呼ばれ，適応度に基づく多項分布に従い新個体をサンプリングする方法である．すなわち，個体集団内の各個体は，適応度に比例したサイズのルーレットスロットを持つ．この重み付きルーレットを「回転」させ，出た目に対応する個体を再生させる．こうすることで，適応度に比例した確率で，個体が再生できる．新たな子個体が必要なたびに，ルーレットを回せばよい．適応度としてランキングを用いる場合は，ランキング選択（Ranking Selection）と呼ぶ．また，個体集団から任意の二個体をランダムに復元抽出し，適応度の高い方の個体を再生させるトーナメント選択（Tournament Selection）もよく用いられる．

上述のような再生操作では，個体集団の平均的な適応度の向上は保証されるが，例えば，最も適応度の高い個体が次世代に生き残る保証はない．そこで，このようなエリートな個体を特別扱いし，無条件に次世代に複製することをエリート戦略（Elitist Strategy）と呼ぶ．

2-6-4 進化的アルゴリズムにおける最適化プロセス

進化的アルゴリズムにおける最適化プロセスの模式図を**図 6・2** に表す．このようなプロセスが一定の世代数繰り返される間に，遺伝的組み換え操作は，常に新たな個体を生成し続ける．

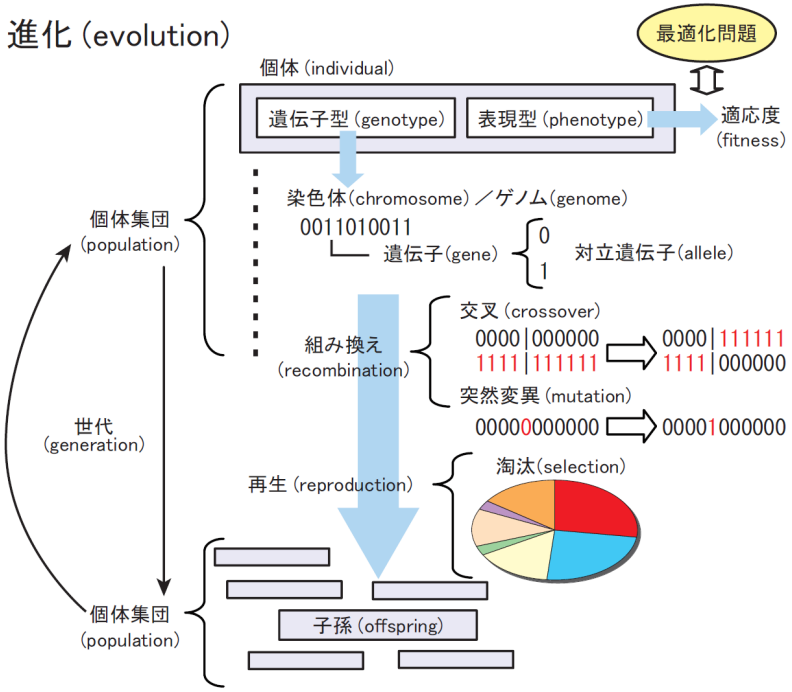

図 6・2　進化的アルゴリズムにおける最適化プロセス

そのなかには，これまでになく適応度の高い個体が生成されることが起こりうる．再生では，そのような優れた個体に着目し，これらを重点的に複製する．これらを繰り返すうちに，個体集団に，適応度の非常に高い個体が出現してくる．これが進化的アルゴリズムの最適化プロセスである．

このように進化的アルゴリズムは，自然界の進化においては非常に長い時間がかかるプロセスを，計算機中で非常に高速にシミュレーションする仕組みであると言える．本節では，解がビット列で表現される最も単純なケースに焦点を当てて説明したが，それ以外の場合，例えば，巡回セールスマン問題や，フローショップスケジューリング問題のような場合，巡回する都市やスケジュールする仕事の順列で解を表現し，順列上の交叉や突然変異を考える必要がある．また，ジョブショップスケジューリング問題では，各機械ごとに仕事の順序を決定する必要があるため，機械の数に対応する複数の仕事の順列で解を表現する．もしくは，各仕事の各機械上での処理（Operation）を頂点とする有向非巡回グラフ（Directed Acyclic Graph）で表現することになる．その場合，グラフ上の交叉や突然変異を定義する必要がある[5]．いずれにせよ，解くべき最適化問題の特性を十分に考慮した，最適な表現と遺伝的オペレータを工夫することが肝要である．

■参考文献

1) D.E. Goldberg：“Genetic Algorithms in Search, Optimization and Machine Learning,”Addison-Wesley Longman Publishing Co., Inc., 1989.
2) K.A. De Jong：“Evolutionary Computation A Unified Approach,”MIT Press, 2006.
3) M. Clerc：“Particle Swarm Optimization,”Iste Publishing Company, 2006.
4) M. Pelikan：“nHierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms,”Springer, 2005.
5) T. Yamada and R. Nakano：“A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems,”Proceedings of The Second PPSN Conference, pp.281-290, 1992.