動的環境下における多戦略学習に基づく学習システム

宮井将之[†] 松浦聰[††] 上原邦昭[†]

従来の学習システムでは，環境が変化しない静的な領域での問題解決を対象としていることが多かった．しかしながら，現実世界は時々刻々と状態が変化し，かつ外界からの影響を受ける動的な領域である．本稿で提案するシステムEbSL（Explanation based Strategy Learning）は，動的な領域を，プレーヤの行動により状態が変化していく時間系列をともなうモデルととらえ，状態を遡りながら行動と状態の関係を説明して学習を行うシステムである．EbSLでは，EBL，失敗に基づく学習，および知識コンパイルという複数の手法を統合した多戦略学習の枠組みを採用している．したがって，問題解決が失敗した場合の経験を通して，知識の選択や行動を制御するための新たな知識を学習できるようになっている．このため，知識を抽出する時点では考慮できなかった，外部のインタラクションを考慮した行動を決定できるという利点がある．

Multistrategy-based Learning System in a Dynamic Environment

MASAYUKI MIYAI,[†] SATOSHI MATSUURA[††] and KUNIAKI UEHARA[†]

Most traditional learning systems deal with static domain which is not affected from environment. On the other hand, real world is dynamic domain whose state changes moment by moment. In this paper, we will propose the new learning algorithm called EbSL (Explanation based Strategy Learning) which can analyze the process of agent's action from the record of game, such as Renju, and can explain cause and effect of agent's action to learn knowledge for problem solving (i.e. strategic knowledge). EbSL adapts framework of multistrategy-based learning which consists of explanation-based learning, failure-driven learning and knowledge compilation. Within this framework, EbSL can get not only knowledge for solving problems but also control knowledge from the set of failure examples. As a result, EbSL can make the decision of action in the interactive and dynamic environment.

1. はじめに

機械学習の分野では，実世界を対象とした問題解決のために，様々な学習システムが提案されている．従来の学習システム，特に演繹的学習に基づくシステム[2),12)]では，与えられた問題の正答に至るルール系列からマクロなルールを生成し，問題解決の実行速度を改善することが研究の主な目的であった．また，システムに与えられる状態は，外界から影響を受けることのない，静的な問題領域を主な対象としてきた．しかしながら，実世界はつねに外界からの影響を受け，状態が変化し続ける動的な領域であるため，ゲームのような問題領域では，他者の存在も考慮に入れた学習が必要となる．たとえば，チェスの盤面の一状態から詰めの概念を学習するといった研究[7)]もなされているが，問題領域そのものは動的な状態を対象としたものではない．

本稿では，動的に変化する問題領域から学習するシステムEbSL（Explanation based Strategy Learning system）を提案する．EbSLは「説明に基づく学習」の枠組み[11)]に基づいているが，学習された知識の適用に誤りがあれば，「失敗からの学習」に基づいて知識の適用を制御するためのメタレベルな知識を学習し，同じ失敗を繰り返さないようにしている．さらに，学習された知識は「知識コンパイル手法」により効率的に利用できるようになっている．このように，EbSLは2つの学習法と知識コンパイル手法を組み合わせた多戦略学習の枠組み[10)]となっており，外界からの影響を受ける動的な問題領域において，問題解決のための効率的な学習を目指したシステムとなっている．

動的に状態が変化する問題領域の例として，本稿では連珠[15)]をとりあげる．連珠を採用した理由として，(1) 競合するプレーヤが存在するため，外界とのイ

† 神戸大学工学部情報知能工学科
Faculty of Engineering, Kobe University
†† 松下電器産業（株）中央研究所
Central Research Laboratories, Matsushita Electric Industrial Co., Ltd.

ンタラクションが存在する動的な領域のモデルの1つと考えられる.

(2) 人工的な問題であるため,形式化を行いやすい.

(3) 学習結果を勝ち負けという客観的な基準により評価できる.

という点があげられる.

2. 学習の概要

説明に基づく学習(Explanation Based Learning, EBL)は,数少ない事例から一般的な概念を学習できるという利点があるが,基本的に対象とする領域は状態が変化しない静的なものである.このため,まったく同じ状態が与えられたときには学習された知識を用いて目標を達成できるが,動的に変化する事例を扱う場合には,以下のような問題が生じる可能性がある.

- 動的な領域では,知識を学習した時点とまったく同じ状態が再現されることはほとんどない.
- 動的な領域では,競合するプレーヤの影響を考慮した知識を学習する必要がある.
- つねに状態が変化するために,時間経過とともに学習される知識の記述が長くなり,計算コストがかかる.

以上の問題を解決するために,本稿ではEBLをいくつかの手法と組み合わせて応用分野を広げることを試みる.まず,1番めの問題に対しては,動的な領域では状態のすべての要素が一度に変化するのではなく,局所的な変化が生じる.このため,状態が変化した箇所のみを発見し,その箇所に関係した部分のみを簡潔に記述する必要がある.本稿では,Epsteinが提案した基本プラン[5]の概念を採り入れ,知識の構成要素としている.さらに,基本プランで表現された局所的に変化した領域を一般化して,まったく同じ状況でなくても適用可能な汎用性のある知識となるようにしている.

2番めの問題に対しては,学習された知識を新しい問題に適用しても,他者からの影響によって失敗することがあるため,失敗からの学習に基づいて知識の適用を制御するためのメタレベルな知識を学習するようにしている.さらに3番めの問題に対しては,知識の肥大化による探索時間の爆発を回避するために,知識コンパイル手法を採り入れて処理時間の効率化をはかるようにしている.

EbSLのシステム構成を図1に示す.EbSLには,あらかじめ初期領域知識としてゲームの基本的な概念を記述した知識が与えられているものとする.また,ゲームの対戦履歴(事例)は時間系列をともなう行動の集合として表現しているため,事例を用いて最終状態(問題解決)に到達するまでの任意の状態を再現可能である.

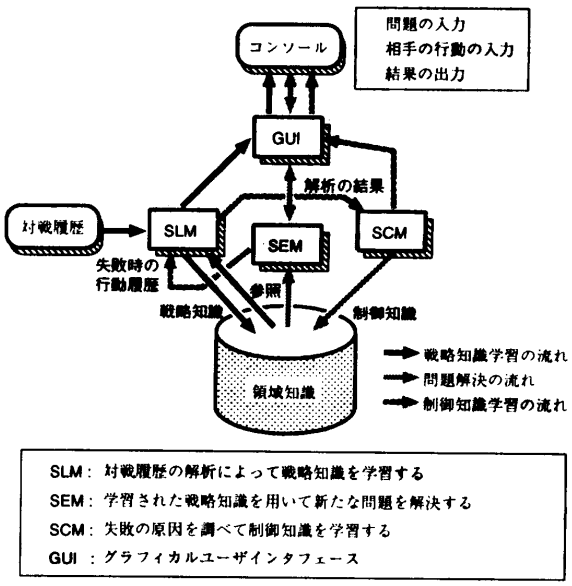

図1 EbSLのシステム構成
Fig. 1 Learning model of EbSL.

まず,学習モジュールSLM(Strategy Learning Module)が,対戦履歴から基本プランで構成された具体的な問題解決のための知識(戦略知識と呼ぶ)を学習する.なお,戦略知識には座標などの状況に依存した値が含まれているため,一般化が行われ汎用性のある知識に変換される.さらに,記述の長い戦略知識は知識コンパイル手法を用いて効率的に適用できる形式に変換される.次に,問題解決モジュールSEM(Strategy Execution Module)が学習された戦略知識を新しい問題に適用することを試みる.もしSEMが目標を達成できない(失敗する)場合には,知識制御モジュールSCM(Strategy Control Module)が,戦略知識の適用を制御するための制約条件を新たな知識(制御知識と呼ぶ)として学習する.

3. SLMによる学習

一般に,「ゲームで勝つ」とは,自分がゴールまでのパスを複数実現した場合であり,そのような状態を必勝状態と呼ぶ.逆にいうと,相手がゴールまでの複数のパスを実現する前にくい止めなければ,自分は負けてしまう.このような行動を阻止しなければならない行動という.SLMは,阻止しなければならない行動を発見するための知識を学習しようと試みる.また,このような知識を戦略知識と呼ぶ.

SLMは,時間系列をともなう行動の集合で再現さ

れる事例を利用して，行動を実行した前後での状態遷移を時間的な因果関係として説明づける．この説明づけの手順を解析と呼び，解析から得られる説明を解析木と呼ぶ．さらに，この説明づけから，どのような場合にどのような行動を実行すれば最終状態に遷移できるかを，戦略知識として学習するようになっている．

3.1 SLMの初期領域知識

SLMには，連珠の初期領域知識として必勝状態の概念の定義を与えている．プレーヤをそれぞれA，Bとし，1ターンでAとBが交互に行動するとき，Aが必勝状態にあるとは，Bがどのような行動を行ったとしても，Aが必ず次のターンで最終状態に到達できる（ゲームに勝利する）状態を意味している．つまり，必勝状態とは

- Bが阻止しなければならない行動をAが複数持っているならば，Aは必勝状態にある．

と定義することができる．さらに戦略知識を得るために必要な概念として，

- Aが最終状態に到達（問題を解決）した場合，Aは必勝状態にある．
- Aが行動Xによって必勝状態になる場合，Bは行動Xを阻止しなければならない．

ということを定義として与えている．

3.2 SLMの学習アルゴリズム

SLMは，ゲームの領域に習熟した人の行動履歴を観察すれば，より効率良く学習を行えるであろうという考え方に基づいて，初期領域知識を参照しながらプレーヤの行動と状態を関係づけて学習しようと試みる．いい換えると，ゲームの必勝状態を知った第三者的な立場から，ゲームの結果を解説するように盤面の状態を逆に遡って解析し，戦略知識を学習する役割を演じている．

SLMの学習アルゴリズムを以下に示す．なお，解析中の「ターンを遡る」という表現は，過去に向かってたどっていく時間的に負の方向を指し，「次のターン」という表現は，ゲームの進行に沿った時間的に正の方向を指すものとする（図2）．

(1) 最終状態T_nの盤面を再現し，状態T_nを最初に発見された必勝状態とする．

(2) ターンを遡り状態T_{n-1}に移行する．ここで，T_{n-1}での行動$n-1$は必勝状態T_nに到達するための行動である．

(3) T_{n-1}から必勝状態T_nに到達可能な行動が複数あるかどうか調べ，T_{n-1}が必勝状態となっているかを判定する．もしT_{n-1}が必勝状態であれば，行動$n-1$は阻止しなければならない行動であるため，戦略知識として保持する．

(4) T_{n-1}がすでに必勝状態となっているならば，さらにターンを遡りT_{n-2}を解析する．以上のようにして，現在の状態が必勝状態である限り，ターンを遡りながら解析を続ける．

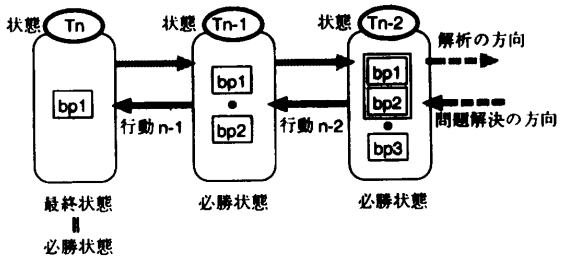

図2 SLMの解析過程
Fig. 2 Analysis process of SLM.

4. SLMによる解析の実行例

4.1 基本プランによる表現記述

SLMでは，連珠の戦略知識を学習するために，基本プランによる知識表現を採用している．基本プランとは，与えられた問題を解決するために必要な最小限の行動からなる順序なし集合である．連珠を例にあげると，石を縦横斜めの任意のラインに連続して5つの石を並べる（五連と呼ぶ）ことができれば，最終状態に到達したことになる．また，プレーヤの行動は石をある座標に置くことに相当しているために，座標を用いて単純に行動を表すことができる．つまり，連珠にとっての基本プランは五連を完成するために必要な行動とその状態からなる集合ということになる．

図3は，連珠の終盤のある状態を切り出したものである．図3の状態では，黒がk6に石を置くことによって四三☆を達成できる．四三の状態では，白がl6に石を置かなければ，ただちに黒にl6を占領されて四連を五連にされてしまう．また，仮に白がl6に石を置いたとしても，黒がl7に石を置けば活三連が達四連☆☆になるため，白がh3とm8のいずれに石を置いても黒は五連を達成できる．すなわち，黒は最終状態（ゴール）へのパスを複数持ち，白が一手ですべてのパスを阻止することはできない必勝状態となっている．

基本プランを用いて図3の必勝状態を表すと，

bp(bp1, 3, ld20, black),
act(bp1, h3), act(bp1, k6), act(bp1, l7),
bp(bp2, 3, ld20, black),

☆ 活三連（止めないと五連になる三連のこと）と四連が同時に存在する状態のこと．
☆☆ 必ず五連になる四連のこと．

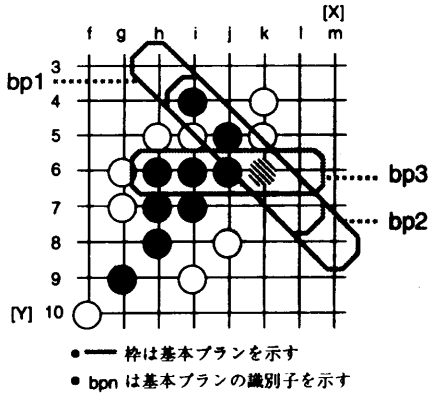

図3 基本プランによる表現
Fig. 3 Expression in terms of basic plan.

act(bp2, k6), act(bp2, l7), act(bp2, m8),
bp(bp3, 2, c6, black),
act(bp3, k6), act(bp3, l6).

となる．ここで，基本プランの状態を示す述語 bp(Bp, Depth, Line, Player) は，各基本プランを表す識別子 Bp，基本プランが最終状態に到達するまでに必要な行動の数（深さと呼ぶ）Depth，基本プランが存在しているライン☆Line，プレーヤ Player が黒か白のいずれであるかを示している．また，基本プランの行動を示す述語 act(Bp, Act) の引数は，基本プラン Bp に含まれる行動（座標）Act を示している．なお以降では，述語とその引数の個数をスラッシュで区切り，bp/4 のような prolog 表記を用いることにする．たとえば，bp/4 は述語名 bp の 4 引数述語であることを表している．

4.2 領域知識と利用可能な概念

基本プランに加えて，SLM では現在の状態が必勝状態であることを示す述語 current_defenseless/3 を導入している．たとえば，current_defenseless(1, [bp1, bp2], black) は，黒の基本プラン bp1 と bp2 の深さはともに 1 であり，現在の状態が必勝状態であることを表している．また，次のターンで必勝状態になることを示す述語 next_defenseless/3 も同様に定義している．さらに，次の必勝状態に到達するための行動を示す述語 next_act(Act, Player) は，次のターンでの行動 Act によりプレーヤ Player が必勝状態になることを表している．

これらの述語間には，図4 に示すような関係がある．すなわち，対戦履歴の解析によって現在の状態が必勝

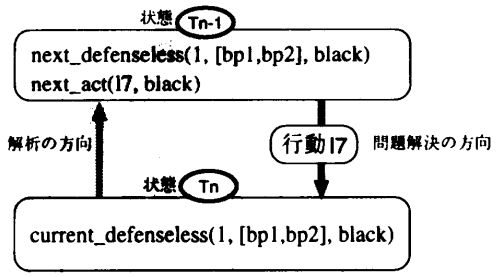

図4 状態を表す述語間の関係
Fig. 4 Relation between predicates which express the state of "Renju".

状態と分かれば（current_defenseless(1, [bp1, bp2], black)），1 つターンを遡った状態では，黒が次のターンでいずれかの深さの最小値が 1 になる基本プラン bp1 と bp2 を持ち，すでに必勝状態になっていること（next_defenseless(1, [bp1, bp2], black)），また，次の黒の行動が l7 であること（next_act(l7, black)）が示されている．したがって，これらの関係を一般化した形で表すと，現在の状態

current_defenseless(Depth, BPlan, Player).

および対戦履歴から，次のターン

next_act(Act, Player).
next_defenseless(Depth, BPlan, Player).

を求めることができるということになる．

これらの述語を用いると，前章で定義した必勝状態の概念の定義は，以下のように記述することができる．

```
current_defenseless(D1, Z, A):-
    must_defense(P1, D1, X, A),
    must_defense(P2, D2, Y, A),
    append(X, Y, Z),
    D1 ≤ D2.                          …(R1)
current_defenseless(0, Z, Player).    …(R2)
must_defense(P, D2, X, A) :-
    next_act(P, A),
    next_defenseless(D1, X, A),
    D2 is D1 + 1.                     …(R3)
```

なお，戦略知識を表す述語 must_defense(P, D, X, A) の引数は，相手から防御するための行動 P，基本プランの深さの最小値 D，基本プラン X，プレーヤ A を示している．

解析を行う対戦履歴として，図5 の状態 T12（最終状態）が与えられるものとする．SLM は，まず盤面状態を基本プランで表現された形に書き換える．次に，領域知識 (R2) を用いて，五連に到達した基本プランと勝利したプレーヤ（黒か白のいずれか）を見つけ出す．図5 では，黒が h3 から l7 にかけて五連を達

☆ 縦のライン：r[X]．横のライン：c[Y]．左から右への斜めのライン：ld[X-Y+15]．右から左への斜めのライン：rd[X+Y-1]．変数 X に X 座標の a を 1，o を 15 とした値を代入し，変数 Y に Y 座標の値を代入して，[ ] 内の式を評価する．

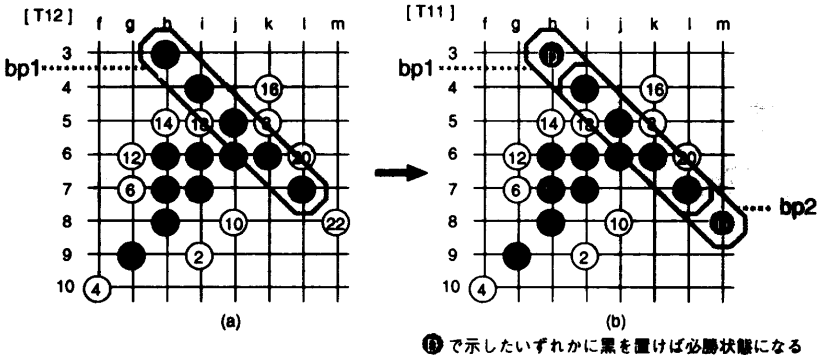

図5 対戦履歴
Fig. 5 History of game.

成しているので，五連を達成した基本プランにラベル bp1 を付ける．

その結果，T12 は必勝状態☆であるので，現在の状態から

next_act(h3, black).
next_defenseless(0, [bp1], black).

という事実が導出される．

一方，これらの next_act/2 と next_defenseless/3 は，対戦履歴の解析結果から導出されたものであるが，問題解決の際には相手の未来の行動や状態を予測できないため，得られた事実をそのまま一般化した知識を適用しても，予想する状態になるとは限らない．たとえば，仮に上記の next_defenseless/3 および next_act/2 を含んだ戦略知識を適用しても，相手の妨害によって実際に深さが 1 になる基本プラン bp1, bp2 を実現できるかどうかの保証はない．

しかしながら，必勝状態から 1 つ遡ったターンには，次の行動でプランを必勝状態に到達させる原因となる座標（行動）が必ず含まれているために，それを現時点での基本プランの位置関係による表現を用いて説明すれば，阻止しなければならない行動，つまり戦略知識を学習することができる．

具体的には，next_defenseless/3 と next_act/2 が得られると（図 6 参照），next_defenseless/3 の第 2 引数の基本プランのリスト [bp1, bp2, ···] を参照して，可能性のあるすべての基本プラン bp/4 および act/2 を求めることになる．なお，bp/4 の第 1 引数は，いずれかの基本プランの深さの最小値に 1 を加えたもの（1 つターンを遡っているため深さが 1 増える）に相当している．また，act/2 の第 2 引数は next_act の第 1 引数から得られるものである．

対戦履歴を 1 ターン遡り，状態 T11 の解析に移る

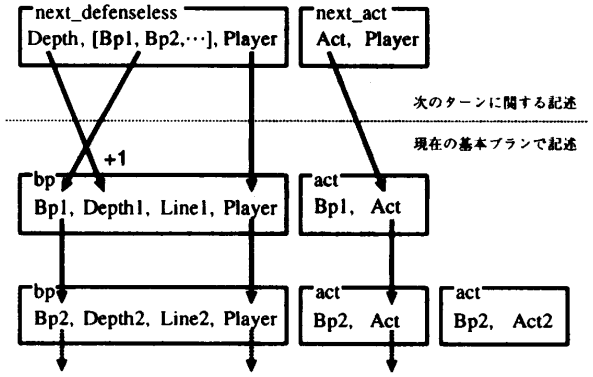

図6 述語の書換え
Fig. 6 Transformation of predicates.

と，T12 で追加された 2 つの事実と領域知識 (R3) から，以下の部分解析木

must_defense(h3, 1, [bp1], black):-
next_act(h3, black),
next_defenseless(0, [bp1], black).

が導出される．この部分解析木は，「次の先手（黒）の行動 h3 により基本プラン bp1 が五連に到達するので，先手の bp1 の行動 h3 は後手（白）にとって阻止しなければならない行動である」ことを意味している．また，この部分解析木には，次の行動を表す述語 next_act/2 および次ターンの状態を表す述語 next_defenseless/3 が用いられているので，先ほどと同様の書換えにより，

must_defense(h3, 1, [bp1], black):-
bp(bp1, 1, ld20, black), act(bp1, h3).

という部分解析木が得られる．

以上のようにして学習される基本プランには，具体的な座標や識別子が含まれているため，まったく同一の盤面とはマッチングできるが，その他の場合にはマッチングできないという問題がある．このような状況に依存した情報を基本プランから取り除くために，次の 2 つの指針に基づいて，基本プランの一般化と呼ぶ概

☆ 五連到達状態も必勝状態の 1 つとして定義している．

念を導入する．

(1) 基本プラン bp/4 の深さ以外の引数は状況に依存しているために，引数を変数化して汎用的なプランにする．

(2) 基本プラン bp/4 のいずれにも含まれない行動 act/2 は削除（省略）する．

したがって，上記の部分解析木が一般化されると，

```
must_defense(P, 1, [B], A):-
    bp(B, 1, L, A), act(B, P).          …(R4)
```

が戦略知識として追加される．なお，(R4) は，「深さ 1 の基本プランの行動はただちに阻止しなければならない」ことを意味しており，(R3) をより具体化したものとなっている．すなわち，(R4) の条件部は図 7 に示す 5 つのパターン（四連と呼ばれる）とマッチする戦略知識となっている．

さらに，追加された戦略知識 (R4) を用いて引続き解析を行うと，阻止しなければならない行動 m8 が発見される．このことから，(R1) を満たす解析木（図 8 参照）が得られ，状態 T11 は必勝状態☆であることが説明づけられる．

さらに，状態 T11 に関する事実から

```
next_act(l7, black).
next_defenseless(1, [bp1, bp2], black).
```

という事実が導出されると，SLM は対戦履歴を遡り状態 T10 を解析する．同様にして，T10 ではこの 2 つの事実から部分解析木

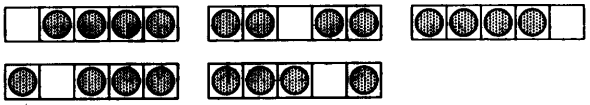

図 7 (R4) の条件部とマッチするパターン

Fig. 7 Pattern "four" which can match the conditional part of strategic knowledge (R4).

```
must_defense(l7, 2, [bp1, bp2], black):-
    next_act(l7, black),
    next_defenseless(1, [bp1, bp2], black).
```

が導出され，

```
must_defense(l7, 2, [bp1, bp2], black):-
    bp(bp1, 2, ld20, black),
    act(bp1, l7), act(bp1, h3),
    bp(bp2, 2, ld20, black),
    act(bp2, l7), act(bp2, m8).
```

と書き換えられる．最終的に，この部分解析木は一般化され，

```
must_defense(A, 2, [B1, B2], D) :-
    bp(B1, 2, D, B), act(B1, A),
    bp(B2, 2, D, B), act(B2, A).        …(R5)
```

となり，戦略知識として追加される．戦略知識 (R5) は，「同一ラインに存在する 2 つの深さ 2 の基本プランに共通する行動は阻止しなければならない」ことを示している．以上の過程を繰り返して，SLM は最終的に必勝状態を達成できなくなるまでターンを遡り解析を進行する．

5. 知識のコンパイル

本章では，SLM によって学習された戦略知識を実際に適用するモジュール SEM と，知識コンパイルによる効率的な知識の利用について説明する．SEM は，新たな対戦履歴を受け取ると，与えられた盤面に対して戦略知識を適用させようと試みる．いい換えると，SLM が第三者的な立場から問題解析を行うモジュールであるのに対して，SEM はプレーヤ自身となって実際に問題解決を行うモジュールである．

5.1 戦略知識の実行例

前章で得られた戦略知識 (R4), (R5) を用いて，SEM が詰め連珠の問題（図 9 (a)）を解決する場合について考える．戦略知識 (R5) を用いると，図 9 (b) のよ

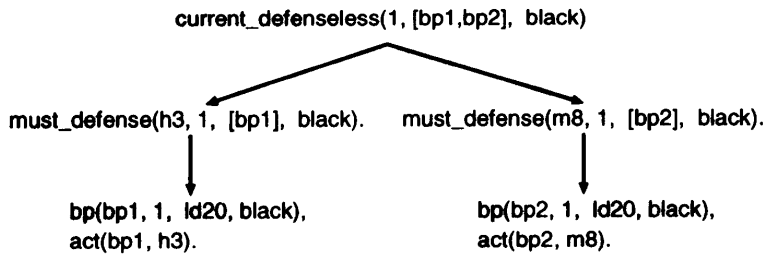

図 8 T11 が必勝状態であることを説明する解析木

Fig. 8 Analysis tree explaining that T11 is in a state of defenseless.

☆ T11 の状態は達四連（必ず五連になる四連）である．

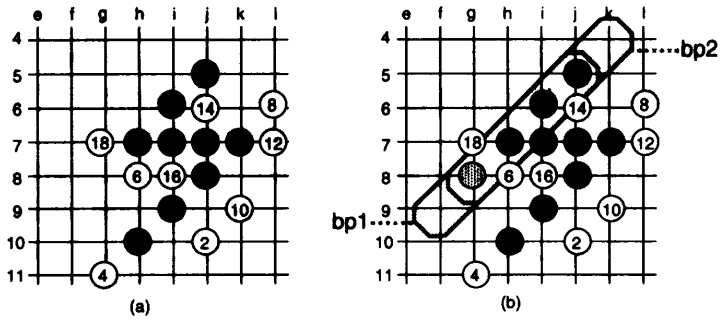

図9 詰め連珠の問題への戦略知識適用
Fig. 9 Application of strategic knowledge to "Tume-Renju" problem.

うに，同一ライン上に存在する2つの深さ2のプランに共通な行動 g8 が発見される．続いて戦略知識 (R4) を用いると，四連を五連にする行動 k4 あるいは f9 が発見され，最終状態に到達する．以上のようにして，SEM は戦略知識を適用しながら，問題解決におけるプレーヤの行動を決定できるようになっている．

この例では，最終状態に近い状況で学習された，条件部の短い戦略知識のみが用いられているが，SLM によって得られる戦略知識には，ターンを遡るにつれて条件部が巨大化し，適用に時間がかかってしまうという欠点がある．この問題を回避する方法については次節で述べる．

5.2 知識コンパイルの適用

抽出された戦略知識を用いた問題解決は，従来からプランニングの世界で研究されている手続き的問題解決に相当している．手続き的問題解決では，戦略知識の適用方法を考慮しなければ，探索の爆発のために有限時間でマッチングが終了しないことがある．たとえば，対戦履歴を解析するごとに戦略知識は増加していくため，

(1) 戦略知識が多くなるにつれ，適切な戦略知識を探索するためのコストが増大する．
(2) 条件部が長い戦略知識では，マッチング自体のコストが大きくなる．

という問題が生じる．

探索コストが大きくなる理由は，SLM によって得られる戦略知識では，基本プランを構成している act の羅列を単純に先頭部からマッチングさせているために，いきなり深さが5の基本プランの探索を行うという可能性が残されているためである．一般的に，盤面上に存在する基本プランは深さが浅いほど数が少なく，逆に深いプランはかなりの数となる☆．このため，単純な座標単位の探索では，コストが非常に多くかかることになる．

ここで，人間が盤面からあるパターンを探し出す場合について考えると，座標単位による探索ではなく，いくつかのマクロ化されたライン単位のパターンを組み合わせているものと考えられる．このような考え方に基づいて，以下では戦略知識の条件部によく現れる基本プランの集合を1つのマクロなプランと見なし，戦略知識の利用を効率化することを試みる．このような効率的な戦略知識への変換手法を知識コンパイルと呼ぶ．

本稿では，知識コンパイルを実現するために，縦，横，斜めの同一ライン上に存在する基本プランを1つのグループにして，戦略知識の本体をグループ化する手法を導入する．なお，得られる基本プランのグループをプラングループと呼ぶ．また，得られるプラングループを，その部分記述を構成する基本プランの深さ D と数 N によって sgD-N と表すものとする．たとえば，プラングループ sg3-2 は深さ3の2個の基本プランから構成されていることを意味している．

ここで，新たに述語 link(PointList, Kind, Line, BpList, Player) を導入して，プラングループの概念を用いた戦略知識の記述について説明する．述語 link/5 は「Line 上に Player が複数の基本プラン BpList を持っており，各基本プランに含まれる座標集合 PointList からプラングループ Kind が構成される」ことを示している．なお，引数 Kind は sg2-1 などのプラングループの種類を表す識別子である．また，Player は石が黒か白を表している．たとえば，得られた戦略知識の本体に以下の部分が含まれていたとする．

```
must_defense(…):- …,
    bp(B1, 3, L1, A), act(B1, P1), act(B1, P2),
    bp(B2, 3, L1, A), act(B2, P1), act(B2, P2),
    …,
```

☆ 15 × 15 の盤面において石がまったく置かれていない状態では，深さ5の基本プランは572個存在する．

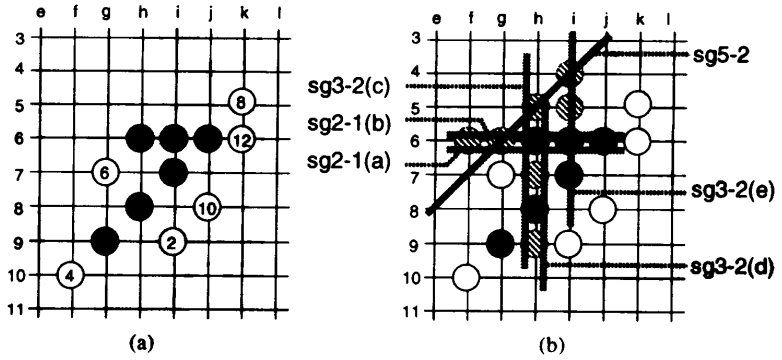

図10 条件部の長い戦略知識の適用例
Fig. 10 Application of strategic knowledge with long conditional part.

この記述の中には，同一ライン上に深さが3の2個の基本プランB1，B2が含まれているので，プラングループsg3-2が構成され，以下のグループ化された表現に書き換えられる．

```
must_defense(…):- …,
    link([P1, P2], 'sg3-2', L, [B1, B2], A),
    …
```

このように，戦略知識の条件部を有限種類のプラングループによってマクロ化しているために，効率良い探索が行えるようになっている．ここで，図10(a)に適用する戦略知識を例にあげる．

```
must_defense(P3, 3, [B1, B2, B3, B4, B5, B6, B7],
  A):-
    bp(B1, 2, L1, B), act(B1, P1),
    bp(B2, 5, L2, B), act(B2, P1), act(B2, P2),
      act(B2, P3), act(B2, P4),
    bp(B3, 5, L2, B), act(B3, P1), act(B3, P2),
      act(B3, P3), act(B3, P4),
    bp(B4, 3, L3, B), act(B4, P2), act(B4, P5),
    bp(B5, 3, L3, B), act(B5, P2), act(B5, P5),
    bp(B6, 3, L4, B), act(B6, P3), act(B6, P6),
    bp(B7, 3, L4, B), act(B7, P3), act(B7, P6).
```

この戦略知識をプラングループを用いた表現に書き換えると，以下のようになる．

```
must_defense(P3, 3, [B1, B2, B3, B4, B5, B6, B7],
  A):-
    link([P1], 'sg2-1', L1, [B1], A),          …(S1)
    link([P2, P5], 'sg3-2', L3, [B4, B5], A),…(S2)
    link([P3, P6], 'sg3-2', L4, [B6, B7], A),…(S3)
    link([P1, P2, P3, P4], 'sg5-2', L2, [B2, B3],
      A).                                      …(S4)
```

上記の戦略知識は，サブゴールとして深さの浅い基本プラングループから順に，最後に深さ5の基本プラングループとなるように構成されている．つまり，プラングループsg2-1，sg3-2，sg5-2が順に作成され，候補の探索もこの順番で行われる．探索の過程で，もし盤面にプラングループを発見できれば，link/5によって記述された事実として蓄えられる．これらの事実は，別の戦略知識をマッチングする際に再利用して，マッチングの効率化を図るようにしている．以下に探索の過程を詳しく示す．

まず，サブゴール(S1)のプラングループsg2-1について探索を行うと，以下に示す2個の事実が発見される☆．

```
link([f6], 'sg2-1', c6, [bp1], black).    (a)
link([g6], 'sg2-1', c6, [bp1], black).    (b)
```

続いて，サブゴール(S2)と(S3)のプラングループsg3-2について探索を行うと，以下に示す3個の事実が発見される．

```
link([h5, h7], 'sg3-2', r8, [bp2, bp3], black).  (c)
link([h7, h9], 'sg3-2', r8, [bp3, bp4], black).  (d)
link([i4, i5], 'sg3-2', r9, [bp5, bp6], black).  (e)
```

ここで，図10(b)に示すように，サブゴール(S4)には，それぞれのプラングループと1つずつ共通した行動が含まれている．このため，同じラインに存在する行動を少なくとも1つ持つプラングループの組合せは{sg2-1(b), sg3-2(c), sg3-2(e)}となる☆☆．したがって，上記の事実を用いてサブゴール(S4)のプラングループsg5-2について探索を行うと，

```
link([g6, h5, i4, f7], 'sg5-2', rd12, [bp6, bp7],
  black).
link([g6, h5, i4, j3], 'sg5-2', rd12, [bp7, bp8],
  black).
```

が発見され，探索が終了する．その結果，戦略知識の行動P3は，サブゴール(S2)または(S3)に属する行動

☆ 基本プランの識別子は見つけた順に付けられる．
☆☆ g6, h5, i4 が同一ライン rd12 上にある．

なので，h5 または i4 が代入されることになる☆．以上のように，深さの浅い基本プランを含むプラングループ間に共通した行動の候補を決定してから，さらに深いプランのマッチングを行っているため，マッチング時に評価する基本プランの数を絞り込むことができるようになっている．

6. 失敗に基づく制御知識の学習

学習された戦略知識が増加するにつれて，最適な戦略知識を効率良く選択するための手法が必要となる．本章では，あるターンで得られる戦略知識と次のターンで得られる戦略知識の間には親子的な強弱関係があることに注目し，戦略知識の制御に用いることを考える．

6.1 戦略知識の強弱関係

図 11 は，対戦履歴を解析して戦略知識 1 と戦略知識 2☆☆が連続的に学習されていく過程を示している．いい換えると，対戦履歴と類似の問題が与えられた場合には，戦略知識 2 → 戦略知識 1 の順で適用すれば，効率良く最終状態に到達できることを示している．これらの関係を階層構造として管理し，戦略知識の制御に利用すれば，効率良く戦略知識を選択できるようになる．

一方，戦略知識 2 は戦略知識 1 の流れを経て学習されたものなので，仮に戦略知識 1 と戦略知識 2 を同時に用いることが可能な場合には，先に戦略知識 1 を用いた方がより早く最終状態に到達できることになる．このような状態が成り立つとき，戦略知識 1 は戦略知識 2 よりも強いという．また，強弱関係を表現した階層構造では，戦略知識と最終状態との間の枝の数が最終状態に到達するまでに必要な手数となっている．いい換えると，基本的にその手数（階層構造の枝の数）以内で最終状態に到達可能であることを示している．したがって，問題解決においては，浅いノードにある戦略知識から階層構造を横型探索し，より少ない手順で最終状態に到達する戦略知識を選ぶことが，強い戦略知識を選択するための効率的な探索となる．

6.2 戦略知識適用の失敗

EBL を用いた学習は，多数の訓練事例を必要とするような「例からの学習」と異なり，領域知識を用いた単一事例からの学習が可能である．しかしながら，SLM によって学習された戦略知識には，相手の戦略・行動に関する情報がまったく含まれていない．そのため，相手から何らかのインタラクションを受け，先に相手が最終状態に到達してしまい，ゲームに「負ける」という可能性がある．本節では，負けた対戦履歴を解析して，同様の失敗を回避するための制御知識を学習するモジュール SCM について説明する．

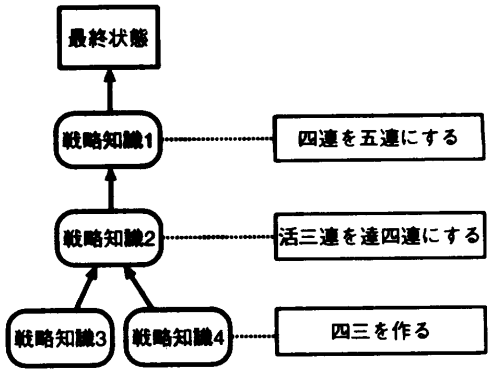

図 11 戦略知識の階層構造
Fig. 11 Hierarchical structure of strategic knowledge.

問題解決における失敗は，帰納的学習における負事例に相当している．正事例のみを満たし負事例を排除する戦略知識を学習するためには，失敗の経験を重ねるごとに，戦略知識の誤った箇所を取り除いて正しい知識を付加する，特殊化と呼ばれる操作が必要となる．しかしながら，単純に特殊化操作を戦略知識に施しても，失敗の経験が多くなれば特殊化の度合が大きくなり，状況に依存しすぎた知識になってしまうという問題が生じる．

このため，以下では，相手からのインタラクションによる失敗は，戦略知識の誤りから生じるのではなく，戦略知識を適用する際の制御知識が不足しているために起こるものと考える．さらに，前述した戦略知識間の強弱関係を制約条件として用いて，適切な戦略知識の選択が行えるような枠組みを検討する．これにより，SLM によって学習された戦略知識には手を加えることなく，完全に独立した制御知識を学習して，戦略知識の適用・管理を容易に行うことができるようになる．

戦略知識の適用に失敗した事例を解析するアルゴリズムを以下に示す．

(1) 負けた事例の盤面の状態を SLM により解析する．
(2) 相手が初めて必勝状態を達成した地点まで遡る．
(3) SCM により，相手が用いた戦略知識と自分が用いた戦略知識との強弱関係を解析し，その結果から得られる制約条件を失敗の回避に利用した制御知識として学習する．

6.3 失敗事例の解析と制御知識の追加

一般に，失敗を認識する時点と失敗の原因となる時

☆ どちらが選ばれても結果は同じになる．
☆☆ 前章の (R4) と (R5) に相当する．

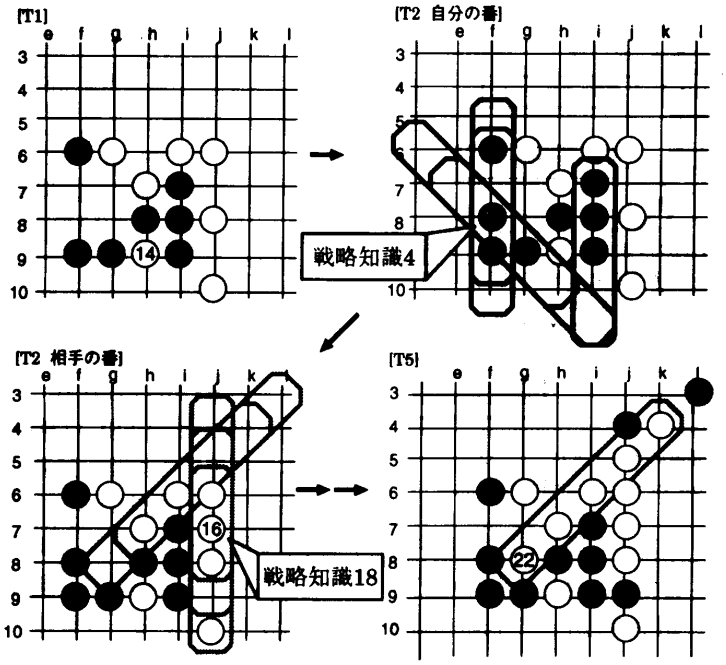

図12 戦略知識の適用に失敗した例
Fig. 12 Failure of game by applying "wrong" strategic knowledge.

点は時間的に離れている[16]．このため，失敗からの学習を行うためには，失敗の原因箇所を正確に見い出す必要がある．また，競合するプレーヤが存在する問題領域では，自分のプランを実行するよりも，相手のプランが最終状態に到達するのを阻止することを優先させなければならない．このような阻止行動と自分のプランを進めるための行動は性質が大きく異なっている．

SEM および SCM では，プラン実行度というプランの性質を表記した属性を定義して，双方の行動にラベルを付けて明確に区別している．つまり，問題解決時に自分のプランに従う行動を起こした場合には + のラベルを付け，自分のプランを進める行動をとれない場合には − のラベルを付けて，行動の性質を記録している．具体的には，必勝状態が継続する限り相手が勝利したときの解析を行い，必勝状態でなくなるまでの相手の行動には + のラベルを付け，必勝状態でなくなったときの行動には − のラベルを付けるようにしている．

図12 は，ターン2において戦略知識の適用を誤った例である．ターン2以降では先手（黒）が後手（白）に追い詰められている．これは黒が白よりも弱い戦略知識を用いたことに原因がある．以降では，この例を用いて失敗の解析を行うことを考える．

図12の失敗を解析したプラン実行度を**表1** に示す．表1には，各ターンにおける双方のプラン実行度と，必勝状態を達成しているかどうかが示されている．これによると，ターン2までは自分のプランを実行しているが，ターン3以降ではプラン実行度が − に変化していること，ターン2以降では相手のプラン実行度が + になり必勝状態を達成していることが分かる．このことから，ターン2が原因となり，攻撃から防御にまわっているために「負け」の状態に陥っていることが分かる．つまり，図12の戦略知識4（まず三を作ってから四三に変形する戦略）と，必勝状態を実現した相手の戦略知識18（まず四を作ってから四三に変形する戦略）には，

戦略知識4 < 戦略知識18

が成立していることになる．ここで，左辺は自分（先手）が用いた戦略知識を指し，右辺は相手（後手）が用いた戦略知識を指している．再び同じ問題が与えられた場合には，学習された戦略知識の強弱関係を用いて，戦略知識4を適用しないようにしている．このような知識を制御知識と呼ぶ．

適用した戦略知識とプラン実行度を用いた戦略知識

表1 失敗事例の解析
Table 1 Analysis of failure example.

状態	1	2	3	4	5
自分のプラン実行度	+	+	−	−	−
相手のプラン実行度	−	+	+	+	+
自分の必勝状態	○	○	○	○	○
相手の必勝状態	×	○	○	○	○
相手の最終状態までの深さ	2	2	1	1	0

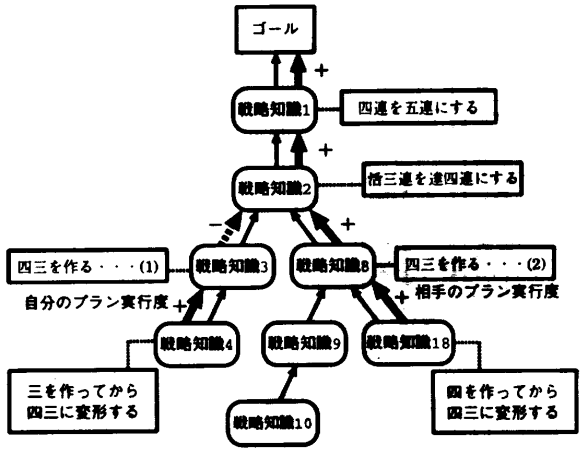

図 13 戦略知識の適用の流れ
Fig. 13 Application flow of strategic knowledge.

の流れを図 13 に示す．ターン 3 以降では，相手が最終状態に向かっているが，自分はプランの実行ができなくなっている．これは，相手の適用した戦略知識 8 が自分の適用しようとした戦略知識 3 よりも強力であり，戦略知識 3 を適用できなかったためである．このため，防御にまわらざるをえない状態となり，最終的に相手が最終状態に到達している．したがって，戦略知識の間の強弱関係

戦略知識 3 < 戦略知識 8

が新たな制御知識として学習される☆．

このように，戦略知識を適用する際には，過去に経験した失敗を再び起こさないように，相手が適用する戦略知識との強弱関係を参照しながら，新たな戦略知識を探索するようにしている．このため，失敗の経験に比例して効率的に戦略知識を適用できるようになっている．

7. EbSL の評価

7.1 失敗からの学習の効果

本節では，EbSL に詰め連珠の問題☆☆を実際に解かせ，システムの評価を行った．まず，5 問ほどの詰め連珠の問題解決事例から戦略知識を学習し，新たな詰め連珠を行うと，表 2 に示す結果が得られた．表 2 に示すように，詰め連珠の最良手を示す模範解答と EbSL の解が一致している割合は多くないが，少ない事例数から問題解決を行う戦略知識を学習できていることが分かる．表 2 の結果のうち，失敗を引き起こした後の行動を検討すると，

- 失敗したあとの再試行で問題を解決できる．
- 失敗からの学習がうまく機能していないため，同じ失敗を繰り返す．

という場合に分けられる．

表 2 失敗からの学習による効果
Table 2 Effect of failure-driven learning.

戦略知識の数	34
問題の数	14
解決した数	7
模範解答と一致した数	3
失敗した後に解決した数	2
再び失敗した数	2

現段階では，EbSL は単純な戦略知識間の強弱関係のみを利用して失敗を回避している．このため，相手が防御しつつ攻撃できる箇所に手を進める場合には，同じ失敗を繰り返してしまうという状況や，実際には手を進めてよい場面で回避行動をとってしまうという状況が生じている．今後は，失敗回避の手法をより洗練して，失敗を引き起こす状況に陥らないための手段を考えねばならない．

ゲームプログラミングの研究では，ゲームの先の局面を読みながら探索を深く行うミニマックス法や α-β 法といった探索アルゴリズムを導入して，上記の問題を回避している．本研究では，EBL および一般化を用いて対戦履歴を遡るという逆向きの学習を主目的としているため，状況に依存した記述を行うと，CBR システムのように事例をほぼそのままの形で格納することと同等になり[3]，事例数の爆発と実際に適用する際のマッチングコストが新たな問題となる可能性がある．このため，5.2 節で述べたプラングループの考え方を用いて，制御知識をより詳細に記述することを検討中である．つまり，失敗した事例を解析し，自分の戦略知識と相手の実行可能なプラングループのうち失敗に関与する部分を抽出し，両者を合わせて失敗を引き起こす状況として制御知識に記述すれば，マッチングコストの問題を抑えつつ，より正確に失敗を回避することが可能になると思われる．

7.2 知識コンパイルの効果

知識コンパイルの効果を確認するために，図 13 の戦略知識 3 と戦略知識 4 を，プラングループを用いずに記述した場合と，プラングループを用いた表現で問題に適用した場合のマッチング時間の比較を表 3 に示す．

表 3 に示すように，知識コンパイルによって分割された戦略知識では，高速なマッチング結果が得られている．これは，戦略知識の条件部を評価する際に，深さの浅い基本プランを含むプラングループから順に

☆ この 2 つの戦略知識はどちらも四三を作るが，記述が少し異なるので区別している．

☆☆ http://www.lemes.se/renju/rifframe.htm 参照．

表3 コンパイルによる探索時間の改善
Table 3 Effect of compilation of strategic knowledge.

	戦略知識3	戦略知識4
コンパイル前	37.2 (s)	992.6 (s)
コンパイル後	7.1 (s)	11.4 (s)

マッチングを行い，候補となる基本プランの数を絞り込んでいるためである．

さらに，ある戦略知識のマッチングに失敗しても，その過程で得られたプラングループの途中経過を，次の候補となる戦略知識のマッチング時に再利用している．このような再利用によってマッチング時の冗長な探索を回避しているため，単体の戦略知識のみならず知識全体の探索時間の効率化が実現されている．

8. 他システムとの比較

8.1 ゲームプログラミング

従来，ゲームに強いプログラムの作成を目的とした多くの研究では，ゲームの盤面状態を先読みするための探索木を作成して，部分的な範囲内で最も良さそうな行動を探索する手法が採用されている[14]．たとえば，Allis の五目並べのプログラム *Victoria* では，threat-space search や proof-number search と呼ばれる効果的な探索法が提案されている[1]．これに対して，本稿で述べた EbSL は，与えられた局面との条件マッチングのみによって行動を決定するリアクティブな戦略知識の学習に限定しており，探索アルゴリズムなどのゲームに強いプログラムについては検討していない．

一方，本研究でとりあげた連珠と同じ領域で，学習やプランニングというコンテキストでなされた研究もすでに多く行われている．たとえば，Elcock らの研究[4]では，五目並べで負けた事例から逆伝播解析（Backtrack Analysis）を用いて，勝ちに結び付くパターンを学習する手法を提案している．これは，SLM における対戦履歴の解析に相当している．また，Findler[6] は学習の過程を 5 レベルに分割し，盤面に存在するパターンとしての知識を段階的に一般化する方法を提唱している．これは SLM における一般化および知識コンパイルと同じような考え方である．さらに，Zahle[13] の手法では，先読みを行う探索木にゲームに負けた対戦履歴のみを例外事例として追加して，同じ失敗を回避する方法を提案している．これは SCM での失敗に基づく学習に相当しているが，制御知識のようなルール形式での知識を学習しているわけではない．

8.2 RLS

EbSL と同様に，メタ知識を追加して知識を修正する学習システムとして RLS[16] がある．RLS は，手続き的知識の一般的な学習を目的としている．すなわち，RLS の学習とは，行った意思決定手続きの何が誤っていたかを特定し，その手続きを修正することである．RLS は，EbSL と同様に，Hayes-Roth の「経験に基づく学習の証明と反証[8]」に準じた「実行-失敗-学習」のサイクルを持っている．つまり，RLS は知識利用，メタ知識生成の複合された枠組みにより，失敗の結果から証明や反証を利用して，持っている知識の修正ができるようになっている．

一方，RLS ではすべての学習が失敗に基づいて行われるので，同一の状況のもとでいくつもの異なるパターンの失敗を繰り返すようにしなければ，妥当な知識が学習できないという問題が残されている．これは，帰納的学習において，負例が少なければ妥当な知識が学習できないという問題と本質的に同等である．これに対して，EbSL は戦略知識を過去の対戦履歴の勝者の行動から抽出している．したがって，得られる戦略知識は対戦履歴のレベルに依存してしまうという問題が残されているが，すべての学習が失敗に依存して行われるわけではないので，効率的な学習が行えるという利点がある．

8.3 CASTLE

CASTLE[9] は，リアクティブな戦略知識（リアクティブルール）の学習を目的としたシステムである．リアクティブルールとは，相手から threat（脅威）☆ がある場合に，複雑なプランニングを行わずに threat を回避する行動を優先するルールのことである．

CASTLE は，プランナによるプランニングによって意思決定を行いながら処理を進行している．また，相手からの threat に対応する必要がある場合には，リアクティブルールを用いるようにしている．リアクティブルールによって計画されたプランが失敗した場合には，なぜプランが失敗したのか，すなわち，なぜ敵のエージェントの行動が成功したのかを説明し，EBL によって新たなリアクティブルールを学習している．しかしながら，CASTLE には，失敗の認識の時点と失敗の原因の時点が離れている場合，同様の失敗を回避するための知識を学習できないという問題が残されている．

8.4 Wyl2

Wyl2[7] は，複数の学習システムを統合して，より効率的な学習を目標としたシステムである．Wyl2 は，従来の EBG，mEBG といった 2 つの学習法に加え

☆ 本稿での概念 must_defence に相当する．

て，これらの生成する概念の不正確さ，不安定さを補うための学習法であるIOE（Induction Over Explanation）を適用している．この3つの学習法を統合して，チェスにおける様々なthreatの概念と概念間の関係が得られることを実証している．

Wyl2もEbSLと同様に，ゲームを問題領域としているが，Wyl2が取り扱っているのは盤面の静的な状態のみである．すなわち，観測される事例にはインタラクション等による状態変化が存在しないことを仮定している．一方，EbSLは対戦過程を対象としており，連続する行動にともなって盤面が変化していく動的な事例の中から目標を達成するための知識を学習できるところが異なっている．

9. おわりに

本稿では，EbSLが適用可能な動的環境下の例として連珠をとりあげているが，すべての動的環境下に適用できるというわけではない．すなわち，EbSLは基本プランの概念を用いて知識表現を行い，EBLによって説明木を構築しているため，満たすべき目標概念は基本プランで表現可能な領域に限定されている．したがって，連珠のような完全情報確定ゲームであって，盤面に石を置いていく形のゲームでは一般的に適用可能である．たとえば，オセロの戦略学習を行う場合には，オセロで有効とされている安定性☆や可動性☆☆などの抽象概念を用いて，勝者の各ターンの状態をゴール達成状態から順に説明づければ，安定性や可動性の高い位置に関する具体的な定義を戦略知識として抽出できる．一方，将棋では目標となる駒（王）が移動するため，目標とする基本プランが毎回変化し，さらに取った駒を置き直すといったことがあるため，基本プランによる戦略知識の記述は困難である．

EbSLの問題としては，失敗を回避する方法として，あらかじめ戦略知識の強弱関係というバイアスを与えている点がある．すなわち，EbSL自身には強弱関係以外に失敗を回避する方法を見い出す機構がないということである．また，戦略知識の適用においても，失敗からの学習を導入しているために，同一の失敗を繰り返さないように，相手よりも弱い戦略知識は選ばなくなっているが，逆に相手よりも強い戦略知識を探索する機構も持っていない．したがって，システム自身が適切なバイアスの移動を行う機構の開発が必要である[7]．いい換えると，システムが「失敗のよりよい概念」の仮説を生成し，それに従って戦略知識の適用に制約を与えるための知識を学習していく枠組みを検討する予定である．

またEbSLでは，他のエージェントからのインタラクションを競合という負のインタラクションに限って考察したが，現実的な問題では協調や協力などの正のインタラクションも存在する．今後は，これらのインタラクションについても考慮し，意思決定支援などのより現実的な問題解決への適用についても検討しなければならない．

☆ 石の反転されにくさ．盤の角のマスでは安定性が高い．

☆☆ 可能な指し手の多さ．これが低いと不利な場所に石を置かざるをえなくなってしまう．

参考文献

1) Allis, L.V., van den Herik, H.J. and Huntjens, M.P.H.: Go-Moku Solved by New Search Techniques, *Proc. 1993 Fall Symposium on Intelligent Games: Planning and Learning*, pp.1–9 (1993).
2) Anderson, J.R.: Plan Abstraction Based on Operator Generalization, *Proc. AAAI-88*, pp.100–104 (1988).
3) Atkeson, C.: Memory-Based Approaches to Learning to Play Games, *Proc. 1993 Fall Symposium on Intelligent Games: Planning and Learning*, pp.101–105 (1993).
4) Elcock, E.W. and Murray, A.M.: Experiments with a Learning Component in a Go-Moku-Playing Program, *Computer Games II*, Levy, D.N.L. (Ed.), pp.273–290, Springer-Verlag (1988).
5) Epstein, S.L.: Learning Plans for Competitive Domains, *Proc. 8th Int. Conf. on Machine Learning*, pp.190–197 (1990).
6) Findler, N.V.: Some New Approaches to Machine Learning, *Computer Games II*, Levy, D.N.L. (Ed.), pp.304–324, Springer-Verlag (1988).
7) Flann, N.S. and Dietterich, T.G.: A Study of Explanation-Based Methods for Inductive Learning, *Machine Learning*, Vol.4, No.2, pp.187–266 (1989).
8) Hayes-Roth, F.: Using Proofs and Refutations to Learn from Experience, *Machine Learning, An Artificial Intelligence Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (Eds.), pp.221–240, Morgan Kaufmann (1983).
9) Krulwich, B.: Learning from Deliberated Reactivity, *Proc. 8th Int. Conf. on Machine Learning*, pp.318–322 (1991).
10) Michalski, R.S. and Tecuci, G. (Eds.): *Machine Learning: A Multistrategy Approach*, Vol.IV, Morgan Kaufmann (1994).
11) Mitchell, T.M., Keller, R. and Kedar-Cabelli,

S.: Explanation-Based Generalization: A Unifying View, *Machine Learning*, Vol.1, No.1, pp.47–80 (1986).
12) Yoo, J. and Fisher, D.: Concept Formation over Explanations and Problem-Solving Experience, *Proc. IJCAI-91*, pp.630–636 (1991).
13) Zahle, T.U.: A Program of Master-Strength to Play Five-in-a-Row, *Computer Games II*, Levy, D.N.L. (Ed.), pp.325–338, Springer-Verlag (1988).
14) 松原 仁：最近のゲームプログラミング研究の動向，人工知能学会誌，Vol.10, No.6, pp.835–845 (1995).
15) 新井華石：連珠必勝法，虹有社 (1979).
16) 田島守彦，実近憲昭：宣言的知識の利用によるRLSの拡張，情報処理学会論文誌，Vol.34, No.5, pp.820–830 (1993).

(平成7年12月8日受付)
(平成8年12月5日採録)

宮井 将之（学生会員）

昭和47年生．平成7年神戸大学工学部システム工学科卒業．現在，同大学院自然科学研究科情報知能工学専攻博士前期課程に在学中．主に人工知能，機械学習の研究に携わる．

松浦 聰（正会員）

昭和45年生．平成4年神戸大学工学部システム工学科卒業．平成6年同大学院工学研究科システム工学専攻修士課程修了．同年松下電器産業（株）入社．以来，主にヒューマンインタフェースの研究に従事．現在，同社中央研究所在籍．

上原 邦昭（正会員）

昭和29年生．昭和53年大阪大学基礎工学部情報工学科卒業．昭和58年同大学院博士後期課程単位取得退学．大阪大学産業科学研究所助手，講師を経て，平成2年神戸大学工学部情報知能工学科助教授．平成元年～2年Oregon State University, Visiting Assistant Professor．平成6～8年神戸大学総合情報処理センター副センター長．工学博士．人工知能，特に機械学習，マルチメディアデータベース，自然言語によるヒューマンインタフェースの研究に従事．1990年度人工知能学会研究奨励賞授賞．人工知能学会，電子情報通信学会，計量国語学会，日本ソフトウェア科学会，システム制御情報学会各会員．