複製と削除の機構を用いた自律的学習機械 —DANDELION†

渡辺俊典†† 佐々木浩二†† 井原廣一††

環境の状況に対して適切な行動を行うことは知的システムの基本的機能であるが，前提として状況と行動との対応関係，認知心理学の言葉をかりれば刺激・反応 (*SR*) 図式の形成が必要である．*SR* 図式の自律的形成を目的として従来さまざまの学習機械が提案されたが，環境の性質，検出器，行動要素が経時的に変化する場合や，行動要素が未知パラメータを含み，パラメータ値自体の最適値を学習する必要のある場合には必ずしもうまく動作しない．この問題に対処するために状況部と行動部からなるコードによる *SR* 関係の記述，コードへの複製能力の付与，特性評点によるコードの淘汰等の手段を利用した *SR* 図式の自律的・適応的学習方式を提案し，計算機シミュレーションによって機能を確認した．

1. まえがき

環境の状況により適切な行動を選択するという知的システムの基礎機能の実現には，当システムの内部に状況と行動との対応関係，認知心理学において刺激・反応 (*SR*) 図式と呼ばれるもの[1]，を形成しておく必要がある．*SR* 図式の自律的形成を目的としてパーセプトロンを始め，いくつかの方式が提案されてきたが[2)~7)]，環境，検出器，行動要素が経時的に変化したり，行動に関連したパラメータの最適値自体を学習する必要がある場合等についてはまだ完全な学習アルゴリズムは得られていない．本論文では，これらの困難な条件のもとで自律的に *SR* 図式を形成する一方式を提案する．

本論文に先立って非線形最適化システムの可用度向上のために，解くべき問題の特徴に応じて解法およびそのパラメータを選択するシステムについて論じ，両者の間の対応関係，すなわち *SR* 図式の半自動的形成のために，問題の特徴抽出結果を条件部とし解法連鎖を行動部とするコードによる *SR* 関係の表現，未経験状況に対処するためのコードを生成する母コードの導入，複製や削除を通じてのコードの淘汰等を特徴とする学習方式を提案した[8)]．

しかしながら，先の提案においては刺激 (*S*) 集合を事前に人為的に分割しておき，これを反応 (*R*) 集合の要素で事後的に埋めてゆくという便宜的な方法をとった．本論文ではこの点を改善し，*S* 集合の自律的分割能力を有するとともに，環境，検出器，行動要素などの経時的変化への適応性を有する自律的学習機構を提案する．

2. 構　想

2.1 課題の考察

SR 図式の自律的形成に際しての問題点を考察するために**図1**のシステム (20) を考える．システムは環境 (10) からの刺激を検出し (30)，*SR* 図式 (40) をもとに環境に働きかける (50)．例としてパターン認識機能をもつ加工機械 (20) を考える．素材 (10) の特徴を認識し (30)，加工作業に関する知識 (40) をもとに作業手続き群 (50) のなかから適切なものを選択し，これを用いて素材を加工する．本論文の目的は図中 (40) の部分を自律的に形成する機構を実現することである．

図中 (40) は，状況検出情報に対して適切な行動を対応させる機能を果たしており，数学的には次の写像とみなせる．

$$\Psi : K(S) \rightarrow A(P)$$

ここに $K(S)$ は検出情報，$A(P)$ は行動要素の集合であり，S は検出情報のパラメータ依存性，たとえば検出器の特性への依存を表し，P は行動要素のパラメータ依存性，たとえば加工時間等，を表す．よって，(40) の自律的形成問題は，Ψ の自律的形成問題として把えることができる．

Ψ の自律的形成に際しての課題を以下に示す．

(1) Ψ の漸次形成：システム (20) の環境 (10) への働きかけを通じて Ψ は徐々に形成される．

(2) 適応性：環境 (10)，検出器 (30)，行動機能 (50) が経時的に変化する動的環境のなかで Ψ が漸次

† DANDELION—Duplication and Deletion Strategy Which Realizes Autonomous Learner by TOSHINORI WATANABE, KOJI SASAKI and KOICHI IHARA (System Development Laboratory of Hitachi).

†† (株)日立製作所システム開発研究所

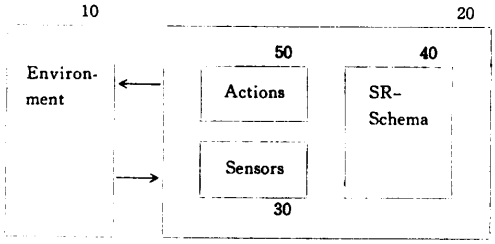


図 1 考察対象の説明
Fig. 1 Simplified representation of antonomous learner.

形成される必要がある．類似経験の繰返しにより行動の選択や行動に関係するパラメータの適正化を行う専門化能力や，Ψ が有効でなくなった場合に Ψ を再構成する可塑性が必要である．

（3） 応用動作：過去に形成した Ψ のなかから，現在の状況に類似のものを想起して再利用する機能が必要である．

2.2 従来技術の考察

写像 Ψ を構成するための代表例としてデシジョンテーブルとパーセプトロンを取りあげる．

（1） デシジョンテーブル[9]：テーブルの列は $K(S)$, $A(P)$ を定義する座標軸であり，各行に Ψ の要素が列挙される．前節の課題(1)に対しては人手によるテーブルの行の追加，(2)に対しては人手による $K(S)$, $A(P)$ 定義列の変更と全行の内容見直しが必要である．専門化，可塑性等の機能はない．課題(3)については，$K(S)$ の定義列要素に“素材の長さが 10 cm 以下”といった表現を導入しておけば，テーブル作成初期に想定した範囲のものについては応用動作可能である．

（2） パーセプトロン[2]：$A(P)$ が比較的少数の要素からなる有限集合であり，$K(S)$ の元がベクトルである場合には，$A(P)$ の元 $\boldsymbol{A}(\boldsymbol{p})$ に対して集合 $K(S)$ 上の判別関数を対応させ，入力 $\boldsymbol{K}(\boldsymbol{s})$ のもとで最大値をとる判別関数に対応する $\boldsymbol{A}(\boldsymbol{p})$ を行動として選択するパーセプトロンにより前節の諸課題に対処できる．課題(1)に対しては，新しい判別関数の追加と判別関数の重み定数の再調整，(2)に対しては，あらかじめ多くの判別関数を保有しておき，$A(P)$ の追加要素に対してそれらを割り当てるとともに，重み定数の調整を常時実施することで，(3)に対しては2点 $\boldsymbol{K}_1(\boldsymbol{s})$, $\boldsymbol{K}_2(\boldsymbol{s})$ が判別関数群による $K(S)$ の分割の同一区画に属する限り，同一の判別関数，いいかえれば行動が選択されるというパーセプトロンの性質によって，それぞれ対処できる．

しかしながら，検出器の追加や欠失により $K(S)$ の特性が経時的変化する場合や，$A(P)$ が小規模の可付番集合によって近似できない場合，たとえば $A(P)$ の元 $\boldsymbol{A}(\boldsymbol{p})$ の実数値パラメータ $\boldsymbol{p}$ の最適値の学習が必要となる場合には(1)(2)に対処することはむずかしい．

2.3 構　想

上述した課題を解決するために，新たに経験した事象が形成中の SR 図式からみて未経験であれば，それを図式に加え，逆に既経験であればそれを引き出して使用する機構を考える．このような系は環境からの情報流入によって常に自己の形相や構造を変化させうる系となっており，熱力学的には解放システム[10),11)]となっている．そのような系を実現するために以下の方法を考える．

（1） コードによる SR 関係の表現：経験した事柄を次のコードの形で記憶する．

$$(\boldsymbol{K}(\boldsymbol{s}), \boldsymbol{A}(\boldsymbol{p})) = (\boldsymbol{K}_1(\boldsymbol{s}_1), \boldsymbol{K}_2(\boldsymbol{s}_2), \cdots, \boldsymbol{K}_n(\boldsymbol{s}_n), \boldsymbol{A}_1(\boldsymbol{p}_1), \boldsymbol{A}_2(\boldsymbol{p}_2), \cdots, \boldsymbol{A}_m(\boldsymbol{p}_m))$$

このコードを必要に応じて蓄積し，課題(1)に対処する．これは，人工知能学におけるプロダクション・ルールと同様の表現である[12)]．

（2） 複製と削除によるコードの淘汰：任意のキーで呼出し可能で，呼び出された場合には乱数を使用して任意の手続き並びを生成できる母コードを用意する．さらに，使用されるつど，自己のコピーを生成し，コピー上のパラメータを乱数によって変更する能力をコードにもたせる．次に，使用したコードの動作特性の良否を評価してコード評点とする．各コードには生成時点を0歳とし，以降コード群へのアクセスが発生するつど，1歳加齢される年齢を与える．蓄積可能なコードの数に制限を与えておき，コードの総数が制限に達した後は評点や年齢によってコード間に競争による淘汰を働かせて劣性コードを削除する．母コードは未経験場面に対する行動の登録すなわち，2.1 節(1)の課題を解決し，複製と評点による淘汰は，使用頻度が低く，増殖機会の少ないコードを追放し，現在の環境に対して有効なコードの増加すなわち Ψ の可塑性を保障する．コード呼び出し時に，特性評点のよいもののうちからランダムに一つを選ぶことにより，コードの特性や評点に種々の外乱が入る場合でも，良好な特性を示す確率の高いコードがしだいに支配的な地位を占めることが可能となる．以上のように，複

表 1　技術課題と解決策
Table 1　Problems and basic concepts.

課題＼方式	デシジョンテーブル	パーセプトロン	構想 コード	構想 複製・削除	構想 テリトリー関数
漸次形成	×	△	○		
適応動作	×	△		○	
環境等の変化	×	△		○	
専門化，可塑性	×	△		○	
応用動作	○	○			○

製・変異形成・淘汰による削除，のメカニズムにより課題(2)に対処する．これは，集団遺伝学における遺伝子集団の適応と進化に関するモデルを足場にした考え方である[13]．

(3)　テリトリ・ポテンシャル関数の導入：コードを呼び出す場合には，現在観測されている情報 $\boldsymbol{K}(s)$ と類似のキーをもつものを対象とすべきである．逆に，新たに生成したコードを追加するために既存のコードを削除する場合には，削除されるコードが守備していた $K(S)$ 上のテリトリを代替守備できるようにしておく必要がある．これらを可能化するには，点 $\boldsymbol{K}(s)$ と各コードとの間に，コードの特性の良否，年齢などに依存した距離を定義しておく必要がある．距離の定義のために，各コードに，そのコードのキー部のベクトル $\boldsymbol{K}(s)$，評点，年齢等をパラメータとした関数を付随させる．これを仮にテリトリ・ポテンシャル関数と呼ぶ．あるコードのテリトリ・ポテンシャル関数は，コードのキー部のベクトル $\boldsymbol{K}(s)$ の周辺にテリトリを生成し，そこに発生する検索キーに対する守備を行う．この考え方は，生物圏における棲み分け現象[14),15)]やプロクセミックス[16)]を足場にしたものである．以上の方法により 2.1 節(3)の課題に対処する．以上の考察を表 1 にまとめる．

3. 構想の実現

3.1　システム構想

図 2(1) に学習機械の構成を示す．

(1)　KB：コード $(\boldsymbol{K}(s), \boldsymbol{A}(p))$ の集合．

(2)　KBM：KB よりコードを呼び出し，右辺 $\boldsymbol{A}(p)$ 部に記入された手続きを実行させる．R & W は RAD エリアの内容の示す KB の番地のコードを KBBUF1 に呼び出し，SAD エリアの内容の示す KB の番地に KBBUF2 のコードを格納する．EXEC は，KBBUF1 上のコードの $\boldsymbol{A}(p)$ 部の要素を逐次 KBCUR に転送し，その内容に基づいて後述する SOLV の要素を呼び出して実行させ，実行後に KBCUR の内容を KBBUF2 に転送する．図中 WKEY, SKEY, RKEY は KB 検索用キーの記憶域でありその上のキーを用いて KB が検索され，上述した SAD, RAD の内容が求められる．

(3)　SOLV：EXEC が呼び出す手続きサブルーチンの集合．環境 E に対して作業を実行する AC 群，実行後にコード特性評価を行う EV 群，継続して呼び出すコードを検索するためのキーを作成する SE 群，キーを用いて KB を検索し SAD, RAD の内容を決定する RE 群のサブルーチンより成る．

3.2　コードの構造

KB 内に，図 2(2)の構造のコードを蓄積する．左辺 $\boldsymbol{K}(s)$ 部，右辺 $\boldsymbol{A}(p)$ 部より成る．

(1)　$\boldsymbol{K}(s)$ 部：$\boldsymbol{K}_m(s)$ は本コードが呼び出されたときの状況パラメータ，J_1, J_2, t, r_1, r_2 は本コードのテリトリ・ポテンシャル関数定義用パラメータである．2 章での $\boldsymbol{K}(s)$ を以下では $\boldsymbol{K}_m(s)$ と再定義し，$J_1 \sim r_2$ を含めたものを新たに $\boldsymbol{K}(s)$ と表す．

(2)　$\boldsymbol{A}(p)$ 部：AC 群サブルーチン呼び出し用の n 個，SE 群，EV 群用の各 1 個，RE 群用の 2 個のフィールドより成る．

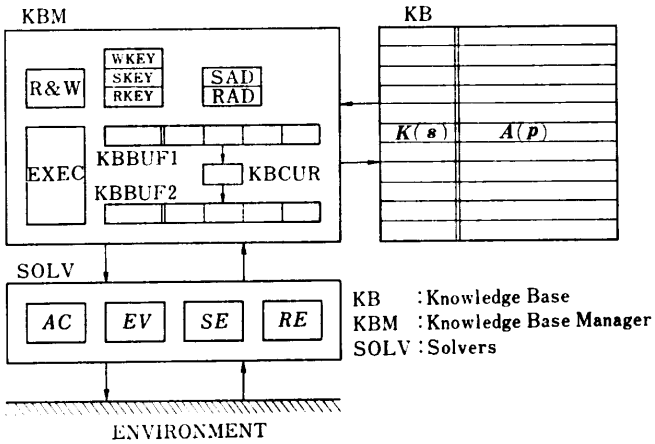


図 2(1)　構想実現の基本方式
Fig. 2(1)　Structure of autonomous learner.

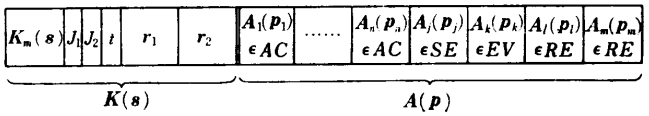


図 2(2)　コードの基本構造
Fig. 2(2)　Code format.

3.3 コードの複製と削除

図2(1)の KB に，(2)のコードを蓄積し KBM に以下の動作を実行させることによりコードの複製と削除が可能となる．

(1) SAD の内容番地に KBBUF2 の内容を収納し，RAD の内容番地のコードを KBBUF1 に読み出す．

(2) KBBUF1 上のコードの右辺 $\boldsymbol{A}(\boldsymbol{p})$ 部を順次 KBCUR に転送し，対応する SOLV の要素の呼び出しと実行を行う．実行後 KBCUR の内容を KBBUF2 に転送する．この操作を全 $\boldsymbol{A}(\boldsymbol{p})$ 要素に対して実行する．

(3) 上記(2)の後，SKEY の内容を KBBUF2 のキー部に転送し(1)に戻る．

コードの構造が図2(2)となっているため，上記(2)の過程は以下のようになる．

まず $\boldsymbol{A}_i(\boldsymbol{p}_i)\in AC$ が解釈，実行される．これらの要素は自己の変異形 $\boldsymbol{A}_i'(\boldsymbol{p}_i')$ を KBCUR 上に生成した後，環境Eに対して $\boldsymbol{A}_i'(\boldsymbol{p}_i')$ を実行するように作成しておく．これにより，KBBUF2 上には，$\boldsymbol{A}_i'(\boldsymbol{p}_i')\in AC$ 配列が生成される．次の $\boldsymbol{A}_j(\boldsymbol{p}_j)\in SE$ により RKEY に継続コード呼出し用キーの $\boldsymbol{K}_m(\boldsymbol{s})$ 部が作られる．次の $\boldsymbol{A}_k(\boldsymbol{p}_k)\in EV$ により，現在実行中のコードの呼出しに使用されたキーの格納されている WKEY の所定部に，実行中のコードのパラメータ $J_1, J_2, t, \boldsymbol{r}_1, \boldsymbol{r}_2$ が記入される（現在実行中のコードの1回前に実行されたコードの右辺の $\boldsymbol{A}_j(\boldsymbol{p}_j)\in SE$ によって RKEY に作られた継続コード呼出し用キーは，同コードの $\boldsymbol{A}_m(\boldsymbol{p}_m)\in RE$ により WKEY に転送されている．したがって，現在実行中のコードの呼出しに用いられたキーは WKEY に記憶されている）．次に $\boldsymbol{A}_l(\boldsymbol{p}_l)\in RE$ により，WKEY の内容を用いた KBBUF2 上のコードの KB への格納先の決定と WKEY 内容の SKEY への転送が行われる．次の $\boldsymbol{A}_m(\boldsymbol{p}_m)\in RE$ により，RKEY 内容の WKEY への転送，WKEY 内容による KB 検索と継続コード呼出し先の決定が行われる．

以上の KBM 動作により，KBBUF2 上には KBBUF1 の変異コピー（$\boldsymbol{K}(\boldsymbol{s})$ 部と $\boldsymbol{A}_i(\boldsymbol{p}_i)\in AC$ 部が異なる）が作られる．この変異コピーは上記(1)により KB に格納される．すなわち，KB 内のコードは使用されるつど，自己の複製を作成し KB の内容を変更させる（KB に空きがあれば複製コードを追加し，空きがなくなった後には劣性コードを削除し，それに入れかわる）．

3.4 *SR* 図式の自律的形成

前節までに述べた構造のシステムにおいて，上述した $\boldsymbol{A}_j(\boldsymbol{p}_j)\in SE$, $\boldsymbol{A}_k(\boldsymbol{p}_k)\in EV$, $\boldsymbol{A}_l(\boldsymbol{p}_l)\in RE$, $\boldsymbol{A}_m(\boldsymbol{p}_m)\in RE$ の機能を適切に定めれば，キー $\boldsymbol{K}_m(\boldsymbol{s})$ の定義域 $\Omega(\boldsymbol{K}_m(\boldsymbol{s}))$ を刺激集合 S とし，行動手続の集合 AC を反応集合 R とする SR 図式を自律的に形成できる学習機械を実現できる．以下にこれを説明する．

(1) テリトリ・ポテンシャル関数

各コードに対して KB からの呼出し，追出し操作に関するテリトリを導入するために，各コードに次の関数組を随伴させる．

$$\{P_1(\boldsymbol{y}, \boldsymbol{K}_m(\boldsymbol{s}), J_1, t, \boldsymbol{r}_1),\ P_2(\boldsymbol{y}, \boldsymbol{K}_m(\boldsymbol{s}), J_2, t, \boldsymbol{r}_2) \mid \boldsymbol{y}\in\Omega(\boldsymbol{K}_m(\boldsymbol{s}))\}$$

P_1 は $\boldsymbol{y}=\boldsymbol{K}_m(\boldsymbol{s})$ で最小値 J_1 を，P_2 は最大値 J_2 をとる．t は年齢，$\boldsymbol{r}_1, \boldsymbol{r}_2$ は P_1, P_2 の開角等を支配する（**図3**）．

$\boldsymbol{A}_l(\boldsymbol{p}_l)\in RE$ によってキー $\boldsymbol{y}^*=\boldsymbol{K}_m^*(\boldsymbol{s})$ で KB よりコードを呼び出す際に，$P_1(\boldsymbol{y}^*, \boldsymbol{K}_m(\boldsymbol{s}), J_1, t, \boldsymbol{r}_1)$ の小さい N 個のコードのなかからランダムに一つ選ぶことにより $\boldsymbol{y}^*$ に類似で特性のよいコードを呼び出すことが可能となる．逆に $\boldsymbol{A}_m(\boldsymbol{p}_m)\in RE$ において，KB へのコード格納先を求める場合，KB に空きがあればそこに，なければ $P_2(\boldsymbol{y}^*, \boldsymbol{K}_m(\boldsymbol{s}), J_2, t, \boldsymbol{r}_2)$ が最大のコードを KB から追い出してそこに収納する．

$N=1$ のとき，キー $\boldsymbol{K}_{m1}(\boldsymbol{s})$, $\boldsymbol{K}_{m2}(\boldsymbol{s})$ をもつ二つのコードの呼出し操作に関するテリトリは $\Omega(\boldsymbol{K}_m(\boldsymbol{s}))$ 上に定義される次の分岐集合に囲まれた領域となる．

$$\{\boldsymbol{y} \mid P_1(\boldsymbol{y}, \boldsymbol{K}_{m1}(\boldsymbol{s}), J_1, t, \boldsymbol{r}_1) = P_2(\boldsymbol{y}, \boldsymbol{K}_{m2}(\boldsymbol{s}), J_1', t_1', \boldsymbol{r}_1'), \boldsymbol{y}\in\Omega(\boldsymbol{K}_m(\boldsymbol{s}))\}$$

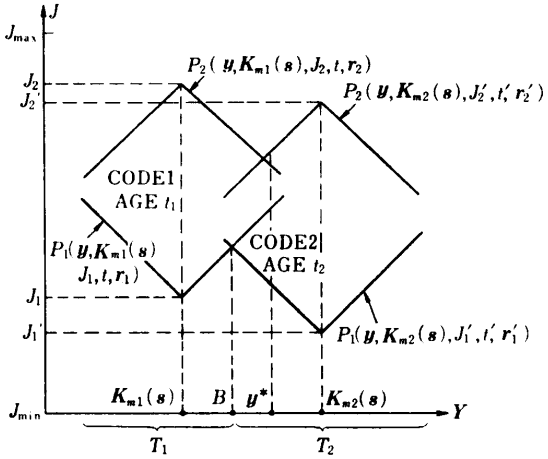


図3 テリトリ・ポテンシャル関数
Fig. 3 Territory potentials of codes.

図3の例では，Y 軸($=\Omega(\boldsymbol{K}_m(\boldsymbol{s}))$)上の点Bが分岐集合であり，左右に二つのテリトリ T_1, T_2 が存在する．同様の方法を関数 P_2 について適用すれば，追出し操作に関するテリトリが定義できる．図示は略す．

次に，関数 P_1, P_2 の具体形を考察する．

P_1 については $\|\boldsymbol{y}-\boldsymbol{K}_m(\boldsymbol{s})\|\to$大，$J_1\to$大（コードの特性が悪いと J_1 大），$t\to$大のとき $P_1\to$大となるべきである．P_2 については $\|\boldsymbol{y}-\boldsymbol{K}_m(\boldsymbol{s})\|\to$小，$J_2\to$大（コードの特性が悪いと J_2 大），$t\to$大のとき $P_2\to$大となるべきである．すなわち $\partial P_1/\partial \boldsymbol{y}>0$, $\partial P_1/\partial J_1>0$, $\partial P_1/\partial t>0$, $\partial P_2/\partial \boldsymbol{y}<0$, $\partial P_2/\partial J_2>0$, $\partial P_2/\partial t>0$. これらの要請を満たす関数の一例として以下の関数を用いる．

$$P_1(\boldsymbol{y}, \boldsymbol{K}_m(\boldsymbol{s}), J_1, t, \boldsymbol{r}_1)$$
$$=J_{\max}-\left\{J_{\max}-\frac{J_{\max}}{1+\left(\frac{J_{\max}}{J_1}-1\right)\exp(-r_{11}t)}\right\}$$
$$\times\exp\{-A(\tilde{\boldsymbol{r}}_1, \boldsymbol{y}-\boldsymbol{K}_m(\boldsymbol{s}))\}$$
$$P_2(\boldsymbol{y}, \boldsymbol{K}_m(\boldsymbol{s}), J_2, t, \boldsymbol{r}_2)$$
$$=\left\{\frac{J_{\max}}{1+\left(\frac{J_{\max}}{J_2}-1\right)\exp(-r_{21}t)}\right\}$$
$$\times\exp\{-A(\tilde{\boldsymbol{r}}_2, \boldsymbol{y}-\boldsymbol{K}_m(\boldsymbol{s}))\}$$

ここで，$0\leqq J_1, J_2\leqq J_{\max}$, $\boldsymbol{r}_1=(r_{11}, \tilde{\boldsymbol{r}}_1)$, $\boldsymbol{r}_2=(r_{21}, \tilde{\boldsymbol{r}}_2)\geqq 0$, $A(\tilde{\boldsymbol{r}}_i, \boldsymbol{y}-\boldsymbol{K}_m(\boldsymbol{s}))$ はベクトル $\boldsymbol{y}-\boldsymbol{K}_m(\boldsymbol{s})$ のノルム化関数であり $A(\tilde{\boldsymbol{r}}_i, 0)=0$ である．$\dim(\boldsymbol{y})=2$, $\boldsymbol{y}=(y_1, y_2)$, $\boldsymbol{K}_m(\boldsymbol{s})=(k_1, k_2)$ のときの2例を下記に示す．

$$A(\tilde{\boldsymbol{r}}_i, \boldsymbol{y}-\boldsymbol{K}_m(\boldsymbol{s}))=\tilde{r}_1|y_1-k_1|+\tilde{r}_2|y_2-k_2|$$
$$A(\tilde{\boldsymbol{r}}_i, \boldsymbol{y}-\boldsymbol{K}_m(\boldsymbol{s}))=\tilde{r}_1 a(y_1-k_1)+\tilde{r}_2 a(y_2-k_2)$$
$$+\tilde{r}_3 a(k_1-y_1)+\tilde{r}_4 a(k_2-y_2)$$

ここに，$a(\alpha)$ は $\alpha>0$ で $=\alpha$, $\alpha\leqq 0$ で $=0$ となる関数である．第1例は，$y_i\gtrless k_i$ に依存せず $\partial P/\partial y_i=\tilde{r}_i$ であるが，第2例は $y_i\gtrless k_i$ によって $\partial P/\partial y_i$ の値を変化させることができる．

（2） パラメータ J_1, J_2, t, $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ の設定

これらは $\boldsymbol{A}_k(\boldsymbol{p}_k)\in EV$ によって設定される．設定の方法は $A(\tilde{\boldsymbol{r}}_i, \boldsymbol{y}-\boldsymbol{K}_m(\boldsymbol{s}))$ の定義等に依存してさまざまのものが考えられるが，ここでは最も簡単な方法を使う．

J_1, J_2：原則として $J_1=J_2$ とする．実行したコードの特性が良好ならば $J_1\to 0$，逆なら $J_1\to J_{\max}$ とする．コードの特性の良し悪しの評価法は問題に依存するので，適用対象に応じて定義する．

t：生成直後のコードについては0とする．ただし，$\boldsymbol{A}_l(\boldsymbol{p}_l)\in RE$ による KB へのコード格納先決定のつど，KB 内の全コードの t を $t+1$ とする．

$\boldsymbol{r}_1, \boldsymbol{r}_2$：コードの老化速度を支配するパラメータ r_{11}, r_{21} については J_1 値を用いて次式で設定する．

$$r_{i1}=\left(\frac{J_1}{J_{\max}}\right)\times(\max(r_{i1})-\min(r_{i1}))+\min(r_{i1})$$

ただし，$i=1,2$ で，$\max(r_{i1})$, $\min(r_{i1})$ は事前に定めた正定数．$J_1\to J_{\max}$ のとき，$r_{i1}\to$大となり老化速度は早められる．次に P_1, P_2 の開角を支配するパラメータ $\tilde{\boldsymbol{r}}_1, \tilde{\boldsymbol{r}}_2$ については，KBBUF1 上のコードの $\tilde{\boldsymbol{r}}_1, \tilde{\boldsymbol{r}}_2$ をそのままコピーして使用する．

（3） 母コードによるコードの生成

$\boldsymbol{A}_j(\boldsymbol{p}_j)\in SE$ によって作成されたキーで KB を検索したとき，KB 内にコードが存在しないとシステム動作が中断する．また，KB 内のコードの数が少ないときには当キーとの類似性の低いキーをもつコードが呼び出される．これらに対処するために，以下のパラメータをもつ母コードを定義する．

$\boldsymbol{K}_m(\boldsymbol{s})$：全要素が '*'．'*' はすべての要素と距離0でマッチングする全能記号．

J_1, J_2：$J_1=J_{\max}$, $J_2=0$

t：任意

$\boldsymbol{r}_1, \boldsymbol{r}_2$：$r_{11}=r_{21}=0$, $\tilde{\boldsymbol{r}}_1=\tilde{\boldsymbol{r}}_1(0)$, $\tilde{\boldsymbol{r}}_2=\tilde{\boldsymbol{r}}_2(0)$

$\boldsymbol{A}_i(\boldsymbol{p}_i)\in AC$：乱数を用いて KBCUR 上に AC の要素の定義情報（名称，パラメータなど）$\boldsymbol{A}_i'(\boldsymbol{p}_i')$ を記入し，この要素を実行する．

以上により，母コードには次の性質が付与される．

$P_1=J_{\max}$, $P_2=0$

$(^\forall\boldsymbol{y}\in\Omega(\boldsymbol{K}_m(\boldsymbol{s}))$, $^\forall t\in[0,\infty])$

すなわち母コードは呼び出されにくく，追い出されにくい．また右辺に上述した $\boldsymbol{A}_i(\boldsymbol{p}_i)$ をもっていることにより KBBUF2 上にコード右辺の AC 部が実際の作業手続きの配列となっているコードが生成される．これより，母コードを当初一本だけ KB に入れておけば検索のたびに KB を自律的に充実させることができる．特性のよい（母コードよりも J_1 したがって P_1 値の小さい）コードが多数 KB に存在すれば母コード呼出しの頻度は低いが，特性の悪い（J_1 したがって P_1 値の大きい）コードが多くなるとこの関係が逆転し，母コード呼出し頻度が増し，KB 内コードの世代交替が早まる．

（4） $K(S)$, $A(P)$ の変化への対処

$K(S)$, $A(P)$ は SR 図式における S と R の集合に対応する．ここでは，これらが変化する場合への対処

法を考察する．

$K(S)$ の変化：最も影響の大きい場合として検出器 $A_j(p_j)\in SE$ で検出できるパラメータの次元が経時的に変化する場合を取りあげる．この場合には，$P_1(y, K_m(s), J_1, t, r_1)$，$P_2(y, K_m(s), J_2, t, r_2)$ において $\dim(y)\geqq\dim(K_m(s))$（＞は検出器追加，＜は欠失のとき）となり，ノルム $A(\tilde{r}_i, y-K_m(s))$ が定義できなくなる．対策として，あらかじめ y および $K_m(s)$ の次元を十分大きく取っておき，観測不能な次元については '＊' マークを埋め，y あるいは $K_m(s)$ の第 k 次元が '＊' のとき，$y-K_m(s)$ の第 k 次元は0とするという演算規定を設けておく．

$A(P)$ の変化：SOLV の部分集合 RE の要素 $A_l(p_l), A_m(p_m)$ は KB 内のコードを検索するものである．コード構造の定義から，これら2要素は必須であるが内部での処理アルゴリズムは変更してもかまわない．このとき KB の環境への適応特性等は変化するが KB 内のコードを人手介入によって修正する必要はない．EV の要素 $A_k(p_k)$ についても同じである．SE の要素変化への対処法については上述した．AC の要素の追加，変更に対処するには AC 内にあらかじめ多数のダミー手続 $A_{id}(p_{id})\in AC$ を定義しておく．ここでダミーは呼び出されても環境Eに対して何ら行動をおこさず即座に処理を終了するサブルーチンである．ダミーの一つに追加したい手続きを埋め込むことで AC への要素追加が容易に実現できる．逆に，既存手続きの内部処理をダミー化することにより AC からの要素削除が実現できる．ただし，評価機能 EV の要素 $A_k(p_k)$ において，ダミー手続きの実行に対してはコード評点 J_1, J_2 を大きく設定する機能をあらかじめもたせておく必要がある．たとえば，コード右辺 $A_i(p_i)\in AC$ がダミーのみのとき，$J_1=J_2\fallingdotseq J_{\max}$ とすることにより，環境Eに対し無動作のコードが KB 内で増大することを防止できる．

4. 機能検証

4.1 機能検証実験計画

（1） 課題の設定

図2（1）の環境Eを2次元平面 $E=\{(x_1, x_2)|0\leqq x_i\leqq 1, i=1,2\}$ とする．SOLV の要素 SE はE内のベクトル $K_m(s)=(k_1, k_2)$ を観測する．システムは $K_m(s)$ を刺激とし，AC の要素を応答として実行する．EV は応答を評価し，コードに評点をつける．E, SE, AC 等に経時変化があっても，適切な SR 図式を自律的に KB 内に形成することがシステムに課せられた課題である．

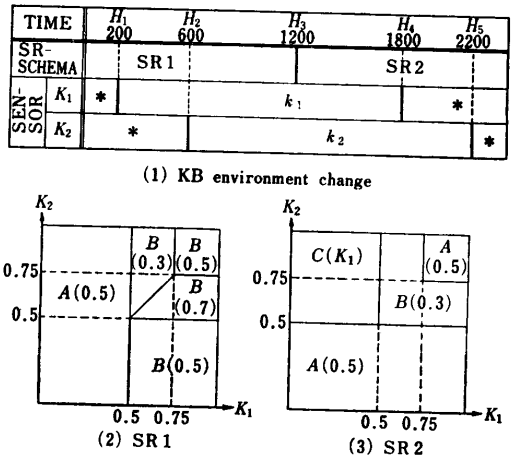


図 4 実験計画

Fig. 4 Experiment design.

（2） KB の環境の変化

KB の環境を**図4**のように変化させる．図4（1）は変化のタイムチャートである．ただし時刻は KB へのアクセス回数で計る．図示したように TIME=H_3 を境に学習すべき SR 図式が SR1 から SR2 に変わる．TIME≦H_1 では $K_m(s)=(*, *)$ すなわち E の双方の次元を観測できない．H_1 で $(k_1, *)$ すなわち第1次元のみ，H_2 で (k_1, k_2) すなわち双方の次元が観測可能となる．H_4 で $(*, k_2)$ すなわち第1次元観測能力の欠失，H_5 で $(*, *)$ すなわち双方の次元の観測能力欠失がおこる．さらに，行動集合も変化する．すなわち，SR2 では SR1 になかった行動 $C(k_1)$ すなわち行動 C をパラメータ k_1（観測情報の第1次元成分）で実行するという手続きが追加され，SR1 に存在した $B(0.5)$, $B(0.7)$ が削除されている．

4.2 システム要素の設定

（1） 環境E：2次元乱数ベクトル $\{(x_1, x_2)|0\leqq x_i\leqq 1, i=1,2\}$ を発生させるサブルーチンで模擬する．

（2） KB：最大 50 コードを蓄積．第1行には母コードがあり，以下は空白の状態が初期状態である．

（3） KBM の構造，動作：3章で述べたとおり．

（4） SOLV の要素：$AC=\{A(p), B(p), C(p), D(0)\}$，**図5**はこれらが KBCUR に呼び出されたときの KBCUR の内容の状態遷移表である．第1行は，KBCUR 上の手続きが $A(p)$ のとき，確率 0.6, 0.2, 0.2 で手続 A, B, C に変異し，またパラメータ p に 0～0.2 の一様乱数が加減されることを示す．$D(0)$

New state / Old state	Procedure name A	B	C	D	Parameter
$A(p)$	0.6	0.2	0.2	0.0	$p\pm R$ (0, 0.2)
$B(p)$	0.2	0.6	0.2	0.0	$p\pm R$ (0, 0.2)
$C(p)$	0.2	0.2	0.6	0.0	$p\pm R$ (0, 0.2)
$D(0)$	0.33	0.33	0.33	0.0	$0+R(0, 1.0)$

注) $R(a,b)$ は a,b 上の一様乱数

図 5 KBCUR 上での手続きの変異
Fig. 5 Mutation of procedures on KBCUR.

Optimal action / Real action	$A(p)$	$B(p)$	$C(p)$
$A(0.5)$	$J_1=J_2=\lvert p-0.5\rvert\times 100$	$J_1=J_2=60$	$J_1=J_2=70$
$B(0.3)$	$J_1=J_2=80$	$J_1=J_2=\lvert p-0.3\rvert\times 100$	$J_1=J_2=80$
$B(0.7)$	$J_1=J_2=80$	$J_1=J_2=\lvert p-0.7\rvert\times 100$	$J_1=J_2=80$
$C(k_1)$	$J_1=J_2=70$	$J_1=J_2=60$	$J_1=J_2=\lvert p-k_1\rvert\times 100$

図 6 EVALUATION 基準
Fig. 6 Evaluation scheme of system behaviour.

行は手続 A, B, C が等確率で設定され，パラメータ p が 0～1.0 の一様乱数で作成されることを示す．$D(0)$ は母コードの右辺の AC 部に使用される．

SE は，上記(1)が作成する (x_1, x_2) にマスクをかけ，KB 検索キー $K_m(s)=(k_1, k_2)$ を作る．$K_m(s)$ は，TIME≦H_1 で，$(*, *)$，H_1<TIME≦H_2 で $(x_1, *)$，H_2<TIME≦H_4 で (x_1, x_2)，H_4<TIME≦H_5 で $(*, x_2)$，H_5≦TIME で $(*, *)$ と時間によって変化する検出器の特性を模擬する手続きからなっている．

EV は，E の出力 (x_1, x_2) に対する正しい応答をその時点での SR 図式より読み取り（たとえば TIME≦H_3 で $(x_1, x_2)=(0.4, 0.5)$ のときは $A(0.5)$ が正答），これと KBM によって実行された AC の要素（ただし，KBCUR 上での変異を受けた後のもの）とを比べ，**図 6** に定めた方法でコードに評点 J_1, J_2 をつける．年齢 t や r_1, r_2 の設定は 3.4 節(2)の方法を使用する．ただし，$J_{max}=99$, $\max(r_{11})=0.2$, $\min(r_{11})=0.02$, $\max(r_{21})=0.08$, $\min(r_{21})=0.02$ とした．関数 P_2 の老化速度は J_1 と強く相関させる必要はないと考え，$\max(r_{11})>\max(r_{21})$ とした．$\tilde{r}_1, \tilde{r}_2$ については下記母コードの項で述べる．

RE は，3.4 節のとおり．関数 P_1 を用いてコードを呼び出すときのパラメータ N は 2 とした．

(5) コードの構造：図 2(2)の構造．ただし，右辺の $A_i(p_i)\in AC$ 部で $n=1$ とした．

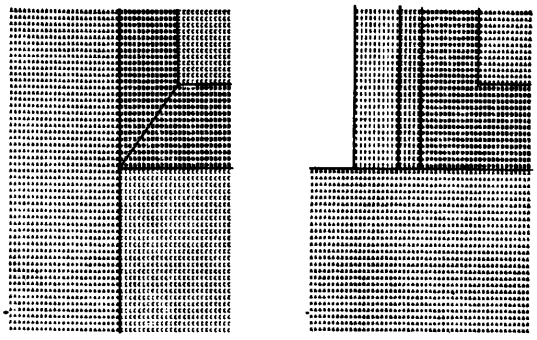
(6) 母コード：3.4 節で述べたものを使用．$K(s)$ 部で $\tilde{r}_1=(20, 20)$, $\tilde{r}_2=(40, 40)$ と設定．$\tilde{r}_2>\tilde{r}_1$ としたのは，コードの追い出し操作に関するテリトリを呼出し操作のものよりも狭くするため．右辺の $A_1(p_1)\in AC$ 部は前出 $D(0)$ を設定した．

4.3 実　験

(1) 実験開始操作：システム起動時に KBM によって KB の第 1 行の母コードを強制的に KBBUF1 に呼び出し，右辺部を実行する．以降，自動的に動作は継続する．

(2) SR 図式の学習様相の観測：KB が図 4(2)(3)の SR 図式を学習する様子をみるため，50 単位時間に 1 回のサイクルで環境 $E\{(x_1, x_2)\mid 0\leqq x_i\leqq 1,\ i=1.2\}$ 上の 50×50 の格子点をキーとして KB を検索し，関数 P_1 の値の小さい N (=2) 個のコードからランダムに 1 個を選び，そのコードの右辺 $A_1(p_1)\in AC$ 部を出力表示させた．ただし，$A_1(p_1)$ の手続名とパラメータ値を同一図に表示するために，$A(0.4\sim 0.6)\rightarrow$'$A$', $A(0.6\sim 0.8)\rightarrow$'+' 等の簡略表示を行った．

図7(1)，(2)は，学習対象である図 4(2)，(3)の $SR1$, $SR2$ をこの方法で表示したもの，(3)は表示記号の説明である．システムの出力がこの図と同一になれば，SR 図式が完全に学習されたことになる．

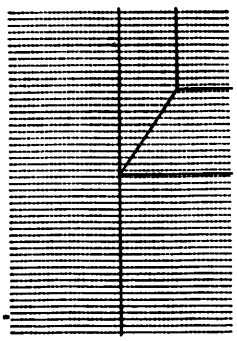


(1) SR 1　　(2) SR 2

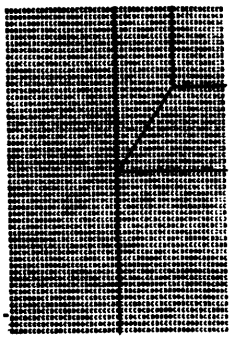
Output	Meaning	Output	Meaning
A	$A(0.4\sim 0.6)$	>	$B(0.8\sim 1.0)$
+	$A(0.6\sim 0.8)$	<	$B(0.0\sim 0.2)$
--	$A(0.2\sim 0.4)$	E	$C(0.4\sim 0.6)$
B	$B(0.2\sim 0.4)$	¬	$C(0.6\sim 0.8)$
C	$B(0.4\sim 0.6)$	|	$C(0.2\sim 0.4)$
D	$B(0.6\sim 0.8)$	·	$D(0)$: mother

(3) Notation

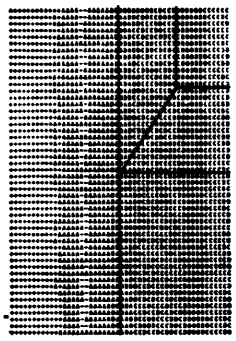
図 7 学習対象 SR 1 および SR 2
Fig. 7 SR schemes to be learned.



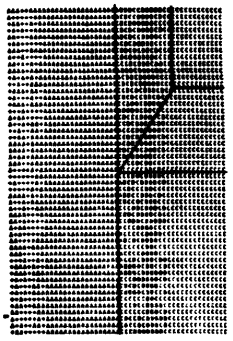
(1) SR 1, TIME=50
$(K_1, K_2)=(*, *)$



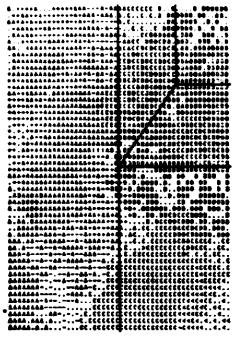
(2) SR 1, TIME=200
$(K_1, K_2)=(*, *)$



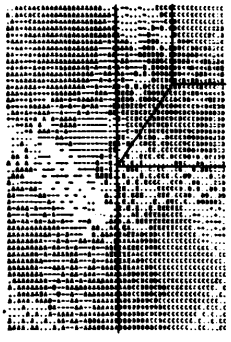
(3) SR 1, TIME=350
$(K_1, K_2)=(k_1, *)$



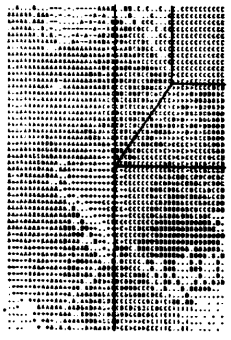
(4) SR 1, TIME=600
$(K_1, K_2)=(k_1, *)$



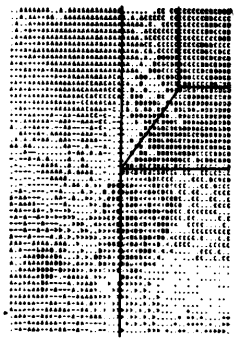
(5) SR 1, TIME=700
$(K_1, K_2)=(k_1, k_2)$



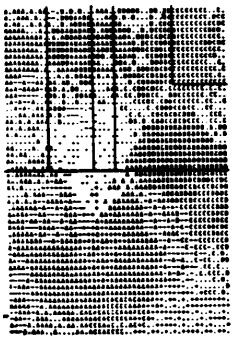
(6) SR 1, TIME=800
$(K_1, K_2)=(k_1, k_2)$



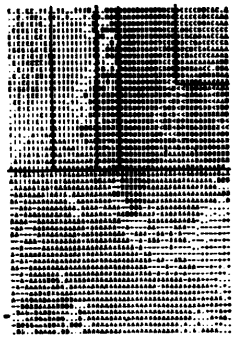
(7) SR 1, TIME=1000
$(K_1, K_2)=(k_1, k_2)$



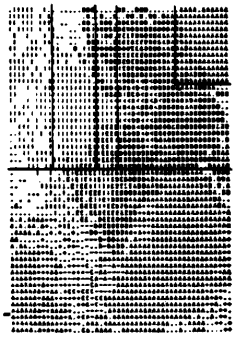
(8) SR 2, TIME=1200
$(K_1, K_2)=(k_1, k_2)$



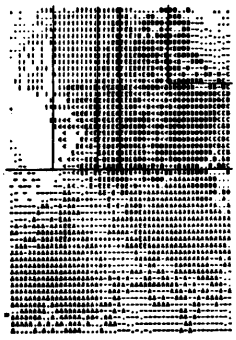
(9) SR 2, TIME=1300
$(K_1, K_2)=(k_1, k_2)$



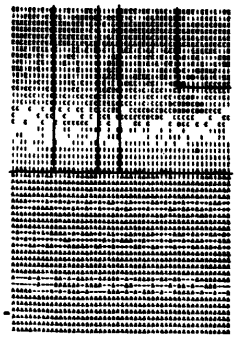
(10) SR 2, TIME=1500
$(K_1, K_2)=(k_1, k_2)$



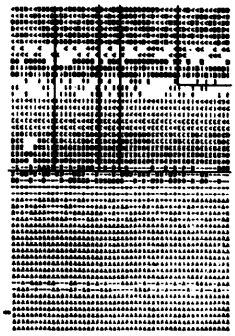
(11) SR 2, TIME=1700
$(K_1, K_2)=(k_1, k_2)$



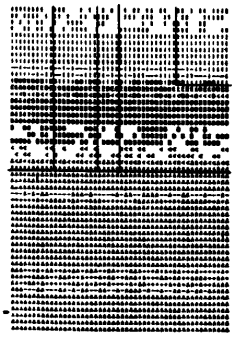
(12) SR 2, TIME=1800
$(K_1, K_2)=(k_1, k_2)$



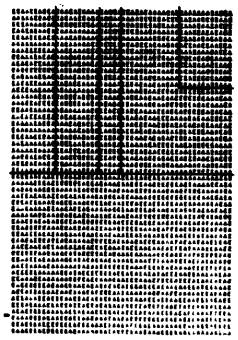
(13) SR 2, TIME=1850
$(K_1, K_2)=(*, k_2)$



(14) SR 2, TIME=2000
$(K_1, K_2)=(*, k_2)$



(15) SR 2, TIME=2200
$(K_1, K_2)=(*, k_2)$



(16) SR 2, TIME=2250
$(K_1, K_2)=(*, *)$

図 8 SR 図式の学習様相
Fig. 8 Process of SR scheme learning.

図8(1)～(16)は実際に得られた出力である．図中(1)～(8)は図4(1)で TIME≦H_3 すなわち学習対象が $SR1$ である場合の出力，(9)～(16)は TIME>H_3 すなわち $SR1$ が $SR2$ に変化した後の出力である．また，(1)(2)では $(K_1, K_2)=(*, *)$ すなわち検出器による環境観測が不可能（図4(1)で TIME≦H_1），(3)(4)では K_1 成分のみ観測可(図4(1)で H_1<TIME≦H_2)，(5)～(12)では K_1, K_2 成分が観測可(図4(1)で H_2<TIME≦H_4)，(13)～(15)では K_2 成分のみ観測可（図4(1)で H_4<TIME≦H_5)，(16)では双方の成分とも観測不可(図4(1)で H_5<TIME)である．

4.4 結果の考察

(1) 写像 Ψ の漸次形成能力：当初1本の母コードのみをもつ KB が，SR 図式を漸次学習する様子が図8(1)～(7)などから確認できる．

(2) 適応能力：図4(1)に示した検出器の変化や SR 図式の変化（その一部に，4.1 節(2)で説明したように行動要素の変化も含まれている）に適応できることが，図8(1)～(16)より確認できる．以下に詳細な考察を行う．

図8(1)，(2)は (K_1, K_2) が観測不能の場合であり過去の経験のうちで最もよかった行動を盲目的に採用しているようである．(3)(4)では $(K_1, K_2)=(k_1, *)$，であり，この検出能力で十分な $SR1$ の $k_1 \leqq 0.5$ の部分では，$A(0.5)$ を正しく学習しているが，$k_1 > 0.5$ では K_2 の検出が不可能なため，種々の応答が未分化のまま混在している．(5)～(8)は $(K_1, K_2)=(k_1, k_2)$ であり，$SR1$ に近いものを学習している．(9)～(12)は，$SR1$ が $SR2$ に変化した後の学習過程であり，SR 図式の変化に適応できることがわかる．$SR2$ に新たに表れた行動 $C(k_1)$ の獲得（(12)左上部），不要となった行動の忘却（(12)右下部）が実現されている．また，従来の学習機械で扱いにくかった非凸形図式の学習が実現されている（(10)右上L型の 'B' 出力，(11)右上と下半の 'A' 出力）．(13)～(15)は $(K_1, K_2)=(*, k_2)$ のときのものであり，K_1 成分の検出能力の欠失に対し，K_2 のみで十分な部分（$k_2 \leqq 0.5$ の $A(0.5)$）の学習結果は保持されているが，その他の部分では学習結果が破壊されたことがわかる．(16)は $(K_1, K_2)=(*, *)$ すなわち，環境Eを観測できなくなった場合のものである．過去の経験の中で得た良好な行動を盲目的に採用しているようにみえる．同一の図式を長く経験させることにより学習が進むこと（専門化）や可塑性があること（上記で説明済み）がわかる．

(3) 応用動作：KB 内の50個のコードによって，図式 $SR1$ や $SR2$ が形成されていることや，たとえば図8(6)右下の 'C' 出力に，コードのテリトリを示すひし形が現れていることなどから，コードが自己のテリトリに対する守備を行っていることがわかる．

5. むすび

5.1 結　論

(1) 環境の状況に対して適切な行動を選択できる知的システムの形成を，両者の間の SR 図式の学習過程として把え，SR 図式の自律的形成にあたっての課題を考察した．

(2) SR 図式の漸次形成，変化への適応，応用動作の可能化などの諸課題に対処するために，状況部と行動部からなる SR 関係の表現，コードへの複製能力の付与，特性評点による劣性コードの削除，テリトリポテンシャルによる S 集合の事後的分割等の手段を用いた自律的学習機械を提案した．

(3) 計算機シミュレーションによって，提案した学習機械が，SR 図式の急変，検出器の追加・欠失，行動要素の追加・欠失，行動要素の未知パラメータ自体の最適値の学習など，従来の学習機械によって実現することの困難であった問題に対処できることを示した．

5.2 今後の課題

提案した学習機械の特性は，r_1, r_2 等のパラメータの初期値や，以降での調整方法に依存して変化する．学習機械のこれらの変数への依存性の分析や，学習性能を上げるためのこれらの変数の選択法について継続した研究が必要である．

謝辞　日頃，ご指導いただいている東京大学教授・伊理正夫博士に，また研究の機会を与えていただいた日立製作所システム開発研究所・川崎淳所長，春名公一部長に深謝します．

参考文献

1) 今田：現代の心理学，岩波全書，岩波書店，東京 (1970).
2) Rosenblatt, F.: *Principles of Neurodynamics*, Spartan, New York (1961).
3) Fukushima, K.: Cognitron: A Self-Organizing Multilayered Neural Network, *Biol. Cybern.*, Vol. 20, pp. 121-136 (1975).
4) Nakano, K.: Associatron—A Model of Associative Memory, *IEEE Trans.*, SMC-2

(1972).
5) Morishita, I. and Yajima, A.: Analysis and Simulation of Networks of Mutually Inhibiting Neurons, *Kybernetik*, Vol. 11, pp. 154-165 (1972).
6) 甘利：神経回路網の数理，産業図書，東京(1978).
7) Slagle, J.R.: *Artificial Intelligence—The Heuristic Programming Approach*, McGraw-Hill, New York (1971).
8) 渡辺：成長能力をもつ解探索システムの研究—解放システム論的アプローチー，情報処理学会論文誌，Vol. 24, No. 1, pp. 57-68 (1983).
9) Hughes, M.L. et al.: *Decision Tables*, MDI Publications (1968).
10) Glansdorff, P. and Prigogine, I.: *Thermodynamic Theory of Structure; Stability and Fluctuation*, Wiley (1971).
11) Bertalanffy, L.V.: *General System Theory*, George Braziller, New York (1968).
12) Winston, P.H.: *Artificial Intelligence*, Addison-Wesley, Reading (1977).
13) Mettler, L.E. and Gregg, T.G.: *Population Genetics and Evolution*, Prentice-Hall, New York (1969).
14) Tinbergen, N.: *The Study of Instinct*, Clarendon Press, Oxford (1969).
15) 今西：生物社会の論理，思索社，東京 (1971).
16) Hall, E.T.: *The Hidden Dimension*, Doubleday & Company, New York (1966).

(昭和58年 4 月 4 日受付)
(昭和58年 5 月10日採録)