人工知能　85－1
(1992.11.19)

部分解を利用した制約充足問題解決

野中　哲
NTTデータ通信株式会社
〒210 神奈川県川崎市幸区堀川町66-2

あらまし

外部から入力された値に応じた解（部分解と呼ぶ）を繰り返し求める問題は、制約充足問題（CSP; Constraint Satisfaction Problem）の１つである。この問題では、同一入力値に対する部分解を再度解き直すことは無駄が多い。そこで、本稿では部分解を蓄積する方法を提案する。具体的には、入力変数を含む制約充足問題について定義した後、部分解の蓄積効果について言及する。最後に定性推論の状態遷移問題に本方法を適用した結果について述べる。

和文キーワード　　制約充足問題、定性推論、定性シミュレーション

A method for Constraint Satisfaction Problem using partial solutions

Satoru Nonaka
NTT DATA COMMUNICATIONS SYSTEMS CORPORATION
66-2 Horikawa-cho, Saiwai-ku, Kawasaki-shi, Kanagawa 210, Japan

Abstract

When we solve Constraint Satisfaction Problem(CSP) repeatedly by changing input values, it is not effective to solve CSP of the same input values again. In this paper, we propose a method of storing the partial solutions those are already solved. We refer to the effect of storing solutions and present the result of adapting this method to the state transition problem based on qualitative reasoning.

英文 key words　　Constraint Satisfaction Problem, Qualitative reasoning, Qualitative simulation

1. はじめに

制約充足問題（CSP; Constraint Satisfaction Problem）は古典的かつ重要な問題であり、人工知能のさまざまな分野で扱われている。定性推論の状態間の遷移を求める問題も、このような制約充足問題の1つである。

状態遷移問題において、すべての状態間の遷移を求めようとすると、それぞれの状態に対して繰り返し制約充足問題を解かなければならない。これに対し、依存関係管理を行うATMS等を用いた方法[1]により、効率良く解く手法が開発されている。しかし、そこで扱う制約充足問題は汎用的な問題であるため、変数のドメインが有限集合である組み合わせ制約充足問題を必ずしも効率良く解くものではない。

本稿では、外部から特定の変数に対する入力が繰り返し行われるような組み合わせ制約充足問題について、部分解を蓄積する方法を提案する。さらに、提案する方法の有効性を、仮想的に生成した制約充足問題により示す。また、応用例として交通挙動推定システム[2][3][4]を挙げ、繰り返し制約充足問題を解く場合に、部分解の蓄積方法が有効であることを示す。

2. 制約充足問題の定義

組み合わせ制約充足問題とは、変数の有限集合、および各変数に対してとりうる値の集合（ドメイン）を与えた場合に、すべての制約を満足するような各変数の値を求める問題である。

- ●変数の集合：$V = \{V_1, V_2, ..., V_n\}$
- ●変数のドメイン：変数 V_i が取り得る値を定めた有限集合 D_i
- ●制約の集合：$C = \{c_1, c_2, ..., c_m\}$
  各制約 c_i は、変数の間の関係である（ex. $c_i = c_i(V_1, V_2, V_3)$ は変数 V_1, V_2, V_3 間の制約）

前記のような条件が与えられた場合に、解はn-組の値で

$(a_1, a_2, ..., a_n) \in D_1 \times D_2 \times ... \times D_n$ となる。

制約充足問題を繰り返し解くことを考慮すると、各変数は以下の3種類に分類される。

- ・入力変数：外部からあらかじめ値が割り当てられている変数（その集合を V_I とする）
- ・出力変数：答えとして出力を要求されている変数（その集合 V_O とする）
- ・中間変数：入力変数、出力変数のどちらでもない変数（その集合を V_M とする）

本稿では組み合わせ制約充足問題のうち、入力変数を少なくとも1つ含む問題を扱うことにする。これに対し、整合ラベリング問題[5]などの制約充足問題は、変数がすべて出力変数である場合の問題と考えることができる。

入力変数を用いた制約充足問題の具体例を以下に記す。

$V_I=\{V1,V2\}, V_O=\{V3\}, V_M=\{V4\}$

$D1=D2=D3=D4=\{a,b\}$

$C=\{c1(V1,V4), c2(V2,V3,V4)\}$

である場合に、例えば入力変数に、

$(V1,V2)=(a,b),(b,a),(b,a),...$

というような入力があり、制約 c1,c2 を満足させる V3 の値を求めることを、入力変数を伴う制約充足問題は意味する。

制約充足問題の等価な表現として制約ネットワークが用いられる。上記例を、制約および変数をそれぞれ形の違うノードで表し、それぞれの間の関係をアークで結んだ制約ネットワークを図1に示す。

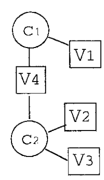

図1：制約ネットワーク

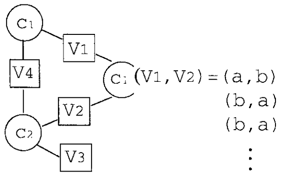

図２：入力変数を考慮した制約充足ネットワーク

これに対し、入力変数間には何らかの制約があり、その制約を満たすような入力しか得られないとし、入力変数間の制約c_Iを示すノードを加えた制約ネットワークを図２に示す。

ただし、この制約c_Iはあらかじめ定まったものではなく、入力値に応じて変化しているという特徴をもっている。

そこで、本稿においてはこの個々の入力変数に対する入力値に対して解いた出力変数の解を部分解と呼び、入力変数を考慮しない全てが出力変数として解いた場合の解と区別する。

3. 部分解の蓄積

入力変数を含む制約充足問題を入力変数の値を変えて繰り返し解く場合に、入力変数値が同一であるものに対し、再度同一問題を解くことは非効率的である。

一方、入力変数のドメインをもとに全ての入力変数に対する入力値の組み合わせを生成し、予め入力値に対応する部分解を蓄積しておくことは、十分な計算機資源と時間が必要となる。特にこの場合、入力変数$V_i,...,V_j$のすべての組み合わせである$|D_i|\times...\times|D_j|$($|D|$は集合の要素の個数を示す)通りのパターンを生成しなければならないが、入力変数の間の制約c_Iにもとづき特定のパターンしか入力されない場合、不要な制約充足処理を行ったことになる。

そこで、これらの問題に対処するため入力変数に対する入力値と対応する部分解（出力変数が取る値）を逐次的に蓄積し、その蓄積結果を利用することを提案する。

図３は部分解の逐次的な蓄積方法を示したものである。

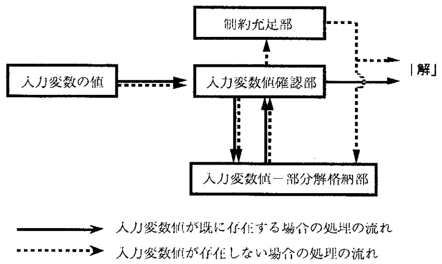

図３：部分解の蓄積

入力変数値確認部は、入力変数の値が既に解かれているかどうかの確認を行う部分である。入力変数の値が入力されると、入力変数値 - 部分解格納部に対してはじめにこの確認を行う。そして、入力値に対応する部分解がある場合は格納部からの値を解として出力する。解かれていない場合は、制約充足部で解き、格納部に部分解を蓄積するとともに、解として出力する。

4. 部分解蓄積による効果

本節では、部分解の蓄積を行うことによる、データの検索に要する時間と、制約充足を行う時間との間の関係について述べる。

入力変数値-部分解格納部において入力変数値j個の入力と対応する部分解が蓄えられている場合を考える。また、入力値として可能性があるすべての組み合わせ数をmとする。入力変数値をキーにして検索する場合に要する時間は一定と考えると、検索に要する時間は、蓄積個数の関数で、$T_{検索}(j)\cdot t_s$と表せる。

一方、入力変数値にあらかじめ値が割り当てられた状態で、制約充足問題を解く場合の制約充足に要する時間を$T_{制約充足}$とする。

入力変数への入力される値の分布が一様であるとすると、部分解の蓄積を用いた制約充足処理時間$T_{蓄積}(j)$はおおよそ次のような値となる。

$$T_{蓄積}(j) = \frac{j}{m} \cdot T_{検索}(j) \cdot ts + \frac{m-j}{m} \cdot (T_{検索}(j) \cdot ts + T_{制約充足})$$

上記の式の第一項および第二項は、入力値に対する検索に要する時間および検索して無かった場合に改めて制約充足を行う時間に、それぞれの起こる確率の積をとったものである。

従って、部分解を蓄積しない場合に対する効果は、$T_{蓄積}(j)$ と $T_{制約充足}$ の比により知ることができる。

$$\frac{T_{蓄積}(j)}{T_{制約充足}} = 1 - \frac{j}{m} + \frac{T_{検索}(j) \cdot ts}{T_{制約充足}} \quad \cdots (1)$$

ここで、制約充足を行う時間 $T_{制約充足}$ が、

$T_{制約充足} = n \cdot tc$

という形で、表されものとする。n は個々の制約を検証（チェック）する回数であり、tc は個々の制約の検証に要する時間である。この場合、制約の検証と検索が同じ時間 ($tc = ts$) であるものとする。さらに、入力変数値-部分解格納部における格納データ形式は、単純に入力値と対応する部分解の値をテーブル形式であり、$T_{検索}(j) = j$ とすると(1)式は次のようになる。

$$\frac{T_{蓄積}(j)}{T_{制約充足}} = 1 - j \cdot \left(\frac{1}{m} - \frac{1}{n}\right) \quad \cdots (2)$$

この関係を図４に示す。

制約の検証回数であるnの値は、入力値により変化するので、蓄積効果もどの入力値に対する制約充足問題を解くかに応じて変化する。従って、n の最小値である n_{min} と 最大値である n_{max} を導入すると、実際は図４における点線で囲まれる領域内で、蓄積効果は変わることになる。

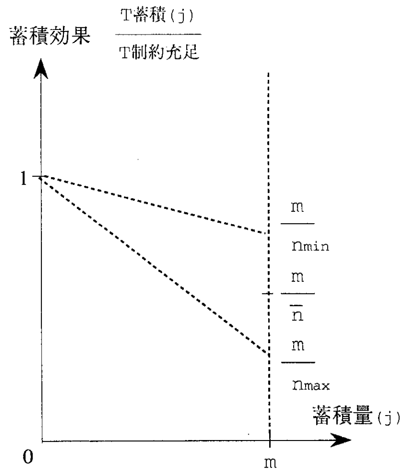

図４：解の蓄積量と蓄積効果の関係

特に、全てのケースにおいて部分解を蓄積した場合が、蓄積しない場合より良いという保証を与えるためには、すべての入力値に対し

$m \leqq n_{min}$

でなければならない。

また、蓄積による効果がどの程度であるかの目安として、

$\frac{m}{\bar{n}}$ （ただし $\bar{n}$ は、nの平均である）

が考えられる。

上記の内容が実際の入力変数を伴う制約充足問題において、どの程度の値となるかを調べるため制約充足問題を生成することにより実験を行った。

[実験]

変数として、入力変数6個、中間変数4個、出力変数6個を選ぶ。すなわち、

$V_I = \{V1, V2, ..., V6\}$,

$V_M = \{V7, V8, V9, V10\}$,

$V_O = \{V11, V12, ..., V16\}$

である。また、各変数のドメインとして

$D_1 = D_2 = ... D_{16} = \{a, b, c\}$

のように、ドメイン数を3に設定する。

制約として2変数間の制約である2項制約を扱い、パラメータとして制約の数と制約の強さの2つを選んだ。制約の数は15、18、21、24の4通りと変えた。その際、入力変数Vi(i=1,2,...,6)と出力変数Vi+10との間には、必ず制約があるものとし、他の制約は、少なくとも制約ネットワークで表現した場合に、それぞれのノードが連結していることを条件にランダムに選んでいる。

また、制約の強さを示すパラメータとして、各制約における許容される値のペアの数（例えば、c1(V1,V7)={(a,b),(b,c),(b,b)}であれば、3である）を3,4,5の3通り変化させて測定した。この各候補となるペアについてもランダムに選んだ。

従って、変数の数は固定とし、パラメータである制約の数と制約の強さを変えた4*3=12通りの実験を、それぞれに対し5つの異なる制約充足問題を生成して行った。

入力変数に対する入力値の生成においては、少なくとも1つの部分解が得られるものとし、制約充足を行った結果、解がないということは起こらないものと仮定した。これは、定性推論の状態遷移問題で、「ある状態kから他の状態へは遷移しない」という判定は、状態kでの変数等の検査により得られるものとし、ここでの制約充足問題の部分解が無いことによって判定するものではないからである。

制約充足の方法により、nの値は影響を及ぼされるが、ここでは木探索方法（バックトラック）により解く方法を採用し、入力変数値を多く含む制約から解くものとした。

制約の数および制約の強さと蓄積効果との関係調べるため、$m / \bar{n}$の値を、パラメータが同一である5つの制約充足問題のうちから、中間の値をとるものを選んで描いたのが図5である。

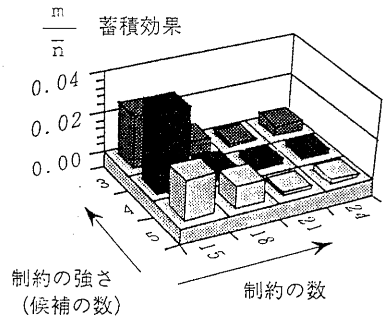

図5：各パラメータと蓄積効果

図5の実験結果は、各パラメータに対する代表的な点であり、各パラメータごとに蓄積効果にばらつきはあるものの、制約の数が少ない場合に$m / \bar{n}$の値は大きいという傾向がみられた。また、実験データでこの$m / \bar{n}$の最大値は0.09程度であった。従って、この値を考えると、多くの場合に蓄積を行わない場合に対し、部分解の蓄積は有効であるといえる。一方、nの最小値であるn_{min}の値は、問題に依存し非常にばらつく場合があり、今回の実験でも、この値がmより小さいケースが1件発生した。

制約充足アルゴリズムと、nの分散に関する十分な検討が、今後望まれる。

5. 交通挙動推定システムへの適用

前節においては、制約充足方法としてバックトラックを用い、仮想的に生成した制約充足問題により、部分解の蓄積方法での有効性について述べた。これは、部分解がある時点で蓄積されていて、その時点で入力変数に対する入力があった場合の効果について述べたものである。実際は、その時点までのすべての蓄積による効果を考慮しないと、状態遷移問題を解いた場合の全体的な効果とならない。

本節では、定性推論の状態遷移問題について

述べる。その後、この状態遷移問題を扱った交通挙動推定システムRADIUSに、部分解の蓄積を用いた場合の効果について述べる。

5.1 定性推論の状態遷移問題

定性推論においては、定性値で表される幾つかの変数により定性状態を表す。そして、ある定性状態から、制約に基づき別の定性状態に遷移していく系の振る舞いをシミュレートすることになる。この時の状態変化の様子を図６に示す。

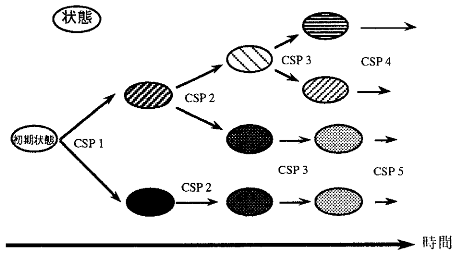

図６：状態変化

ここで、定性値とは、実数軸を有限個の境界標(landmark)によって分割(量子化)し、与えられた量がそのどの区間(あるいは境界標上)にあるかを示したものである。例えば、ある変数 x の境界標が0 のみである場合には、$\{X|X<0\}, \{X|X=0\}, \{X|X>0\}$の３つに分類され定性値$[X]$は、$\{-, 0, +\}$の３状態から値をとることになる。したがって、定性値自体は有限集合から値をとる。一方、定性状態は、系を特徴づける変数がどのような定性値をとっているかにより表される。

ある状態から別の状態への遷移においては、系の状態を表す変数が、変数間の制約によりどのように変化するかを調べる。ここで、状態を表す変数は実際は同一のものであるが、状態kと状態$k+1$の変数の集合をそれぞれV_k, V_{k+1}と別の変数として扱うと、ここでの状態遷移問題は、2.1節での入力変数を伴う制約充足問題（入力変数の集合$V_I=V_k$, 出力変数の集合$V_O=V_{k+1}$）として扱うことが可能である。従って、すべての状態変化や状態変化の履歴を求めことは、入力変数を伴う制約充足問題(図６ではCSPと略記)を、入力変数を変えて繰り返し解くことを意味する。

5.2 交通挙動推定システムを用いた実験

交通挙動推定システムRADIUSは、一交差点に流入する12本の車線の状態により、交差点近辺の道路状態を表している。そして、この道路状態が信号の１周期である140秒間の間にどのように変化していくかを、シミュレートするシステムである。

図７は実際の予測を行う交差点の様子を示したものである。

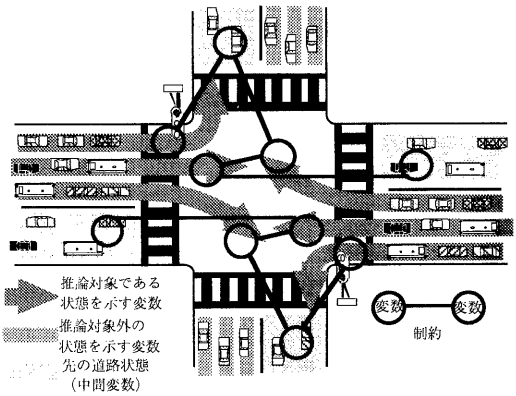

図７：RADIUSでの変数と制約

状態を示す変数はこの交差点に流入する12本の車線であるが、実際の推論において制約充足問題を行うのは、そのうち車が流れることが可能な最大６本の車線（青矢印などの信号の現示により、これ以下となる場合もある）が対象となる。従って、現在の状態から次の状態への遷移を求める場合の制約充足問題において、入力変数、出力変数はそれぞれ６（図７における矢印のついた６本の車線に相当する）である。一方、この12の変数の他に、交差点に流入した車は、この交差点を通り過ぎた先の道路状態により影響を受けるので、中間変数として交差点か

ら流出する４つの車線に相当する４つの変数が考えられる。

また、各変数のドメインは、交通工学の分野において道路の定性的な混雑の度合いを示す６つのサービス水準(混雑度)に対応したものであるが、実際の交差点を観測した知識に基づき、そのうちの３サービス水準にあらかじめ限定している。従って、ドメイン数は３となる。

また、制約としては以下の３つの２項制約がある。

（１）入力変数と出力変数間の間の制約

不連続変化を行わない（現状態がサービス水準Bである場合、次のサービス水準がD等へ急激に変化することがない）。制約の強さは、許容される値のペア数にして6である。

（２）中間変数と出力変数の間の制約

信号を通過した先の道路状態により、通過前の車線の次の状態が規定される。制約の強さは３程度である。

（３）出力変数間の制約

右折車線は対向直進車線の状態の影響を受けるため、その間の関係を規定したもので、制約の強さは６程度である。

この制約充足問題における制約は、図５において、制約の数15、制約の強さ4.6程度に相当する。従って、制約ネットワークの構造と制約の強さの点では、蓄積効果の悪い制約と考えられる。

そこで、このような制約を用いて、問題を繰り返し解く場合の全体的な効果を調べるため、次のような実験を行った。

[実験]

信号の１サイクル（140sec）後である最終状態に達するまでのすべての状態遷移を求める。その場合に、部分解の蓄積を行う場合と、行わない場合の２通りで実行時間の比較を行う。

図８の横軸には、最終状態に達するまでの状態遷移の履歴の異なるものがどれくらいあるか（例えば、初期状態→状態１→最終状態と、初期状態→状態２→最終状態という２通りで同一状態に達する場合には、この値は２としている）をとっている。縦軸は、部分解の蓄積を行わない場合の、最終状態に達するパスの数を基準値１にして、制約充足に要した相対的な時間を縦軸にとったものである。

また、部分解を蓄積する場合には、部分解を全く蓄積していない状態で、初期状態に対する入力を行い、部分解の蓄積は、初期状態から最終状態に達するまでの遷移を求めている間に行ったものである。

RADIUSのインプリメントは、Sun Sparc Station2 上の Quintus Prolog を用いて行っており、部分解の蓄積のインプリメントおよび実行時間の測定においても同様の環境で行った。

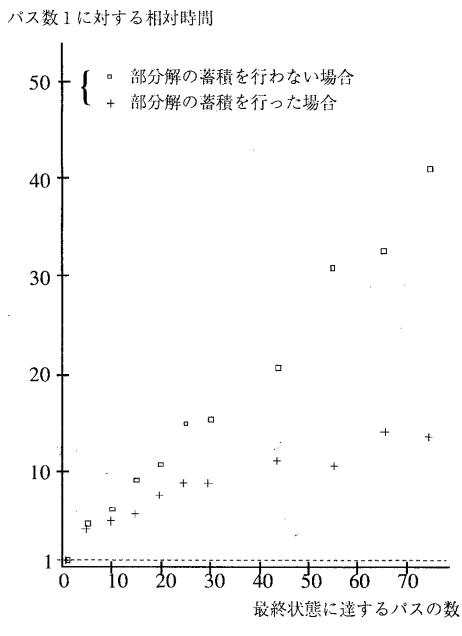

図８：交通挙動推定による実験

図８の実験では、最終状態に達するパスの数が多いほど、制約充足問題を繰り返し解いていると考えられる（ただし、必ずしも線形的に増加するものではない）。従って、部分解の蓄積が繰り返し制約充足問題を解く場合に、特に有

効となることがわかる。最終状態に達するパスの数の平均は40程度あり、およそ2倍程度の効果であった。

交通挙動推定においては、信号の現示パターン（140秒間に4つのstageを取る）により予測を行う車線が異なるため、個々のstageごとに異なる制約充足問題を解くことになる（図9参照）。従って、実際の部分解の蓄積においては、制約充足問題ごとに個別に部分解の蓄積を行っている。

各信号stageにおいては、1から3回程度の状態を経て（すなわち制約充足問題を解いて）最終状態に達している。実験では、各信号現示での制約充足問題における入力値と対応する部分解の蓄積数は、図8の最終状態へ至るパスの数が75の場合においても、10程度であった。

従って、ここでは入力値に対応する検索時間は制約充足に要する時間に比べ事実上無視できると考えられる。部分解の蓄積を用いたことによる全体的な効果は、入力変数に対し蓄積されている入力値と同一のものがどれくらい同じ確率で発生したかに依存している。

RADIUSにおいては、入力変数に入力される値は前段の制約充足問題における出力変数であり、入力パターンの分布と蓄積内容を考慮した上で、部分解蓄積効果の十分な検討が今後の課題となる。

6. まとめ

入力変数に対する入力が繰り返し行われる制約充足問題に対し、入力値に対応する部分解を蓄積する方法を提案した。仮想的に生成した問題の場合、本方法を行わない方法に比べ10倍以上有効である。また、交通挙動推定システムの場合、状態遷移数が多いほど効果があり、平均2倍程度有効である。

参考文献

[1]Forbus, K. D. "The Qualitative Process Enjine", University of Illinois Department of Computer Science Technical Report No. UIUCDCS-R-86-1288, 1986

[2] 杉本勉他"定性推論の交通挙動推定への適用", 第43回情処理学会全国大会, 1991

[3] 野中哲 他 "交通挙動推定への知識処理技術の適用について", 第11回交通工学研究発表会, 1991

[4] 大濱寛樹他"交通状態予測における定量データの定性推論過程への取り込みについて", 第44回情報処理学会全国大会, 1992

[5] 西原清一"整合ラベリング問題と応用",情報処理学会誌, Vol.31. No. 4, pp.500-507, 1991

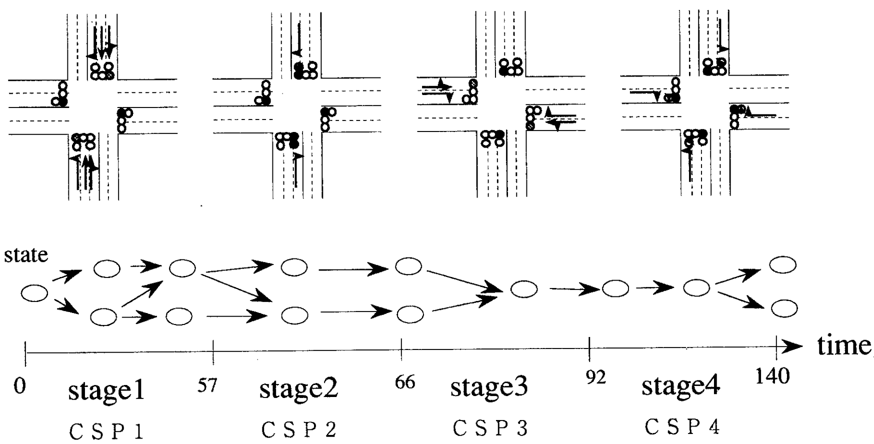

図9：状態の変化と信号の現示