数理解析研究所講究録
第259巻 1976年 25-34

組合せ論の証明とプログラミング
——パリティ検査の一般化——

山梨大学工学部
有澤　誠

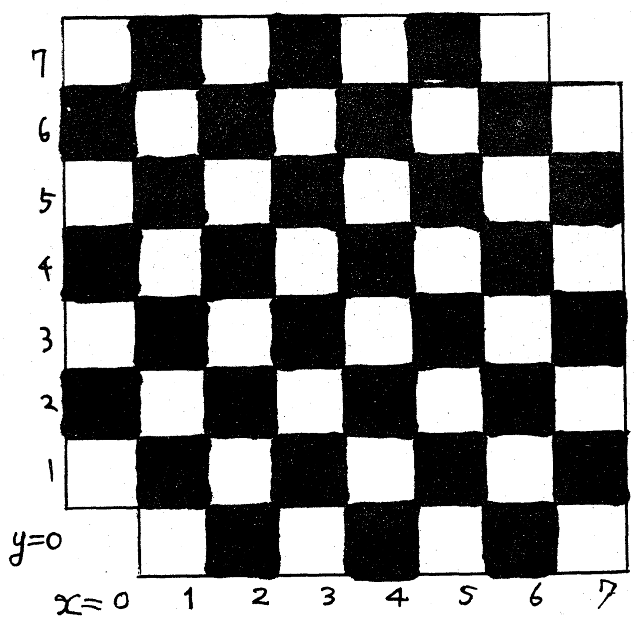
はじめに　数学の証明手法のパターンを分類すると、演繹法、帰納法、背理法などが挙げられる。ここで共通して使用することができる技法のひとつとして、「不変量」に注目してそれが保存される、あるいは保存されない性質を利用することをとりあげる。そのような不変量のひとつとして、組合せ論に現われることの多い、「パリティ」の概念およびパリティの一般化について調べる。特に、組合せ問題をコンピュータを用いて解く場合に、パリティおよびその一般化の考えかたを、どのようにとりいれてゆくべきかについて検討してみることが、本稿の目的である。

平面のパリティおよび一般化

最も有名な問題は、次ページに示すようなチェス盤 8×8 の盤面から、左下隅および右上隅を除いたものを、31個のド

ミノ、□□ この形の図形できっちりおおう問題である．チェス盤の格子は、$x+y \ \underline{\mathrm{mod}}\ 2 = 0$ の位置は白に、また、$x+y \ \underline{\mathrm{mod}}\ 2 = 1$ の位置は黒に、それぞれ図のようにぬりわけられていることから、白の総数32、黒の総数30を数えあげておいて、ドミノ図形でおおうという操作では、必ず白と黒を1つずつおおうという事実を利用して、解が存在しないことを「証明」するわけである．

ここで、図のようなぬりわけの存在は、不可能の証明のためには使用できるけれども、可能なことの証明にはそのままでは使用できない．例として、同じ8×8の盤面から左下隅を除いた、63個のますめを、こんどは21個の □□□ この形の図形できっちりおおう問題を調べてみる．前の問題と同様に考えて、次のように色をぬりわける．以下では、0, 1, 2 … を用いて異る色を表わすことにする．



$$\begin{cases} x-y \ \underline{\mathrm{mod}}\ 3 = 0 \\ x-y \ \underline{\mathrm{mod}}\ 3 = 1 \\ x-y \ \underline{\mathrm{mod}}\ 3 = 2 \end{cases}$$

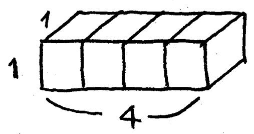
この場合のようすを

図にしたものをあげる．左下隅を含めてかぞえると、0が22個あり、1および2が21個ある．そこで左下隅を除いてしまうと、それぞれ同数ずつとなって、つじつまが合うかのように見える．ところが図形の対称性によって、もし左下隅を除いた盤をおおいつくせるのならば、右下隅を除いた盤もまたおおいつくせるはずである．ところがこの場合は，0が22個、1が20個、2が21個であることから、盤をおおいつくすことはできない．

この例から、つごうのよいぬりわけかたが必ずしも可能であることの証明には使用できないことがわかった．

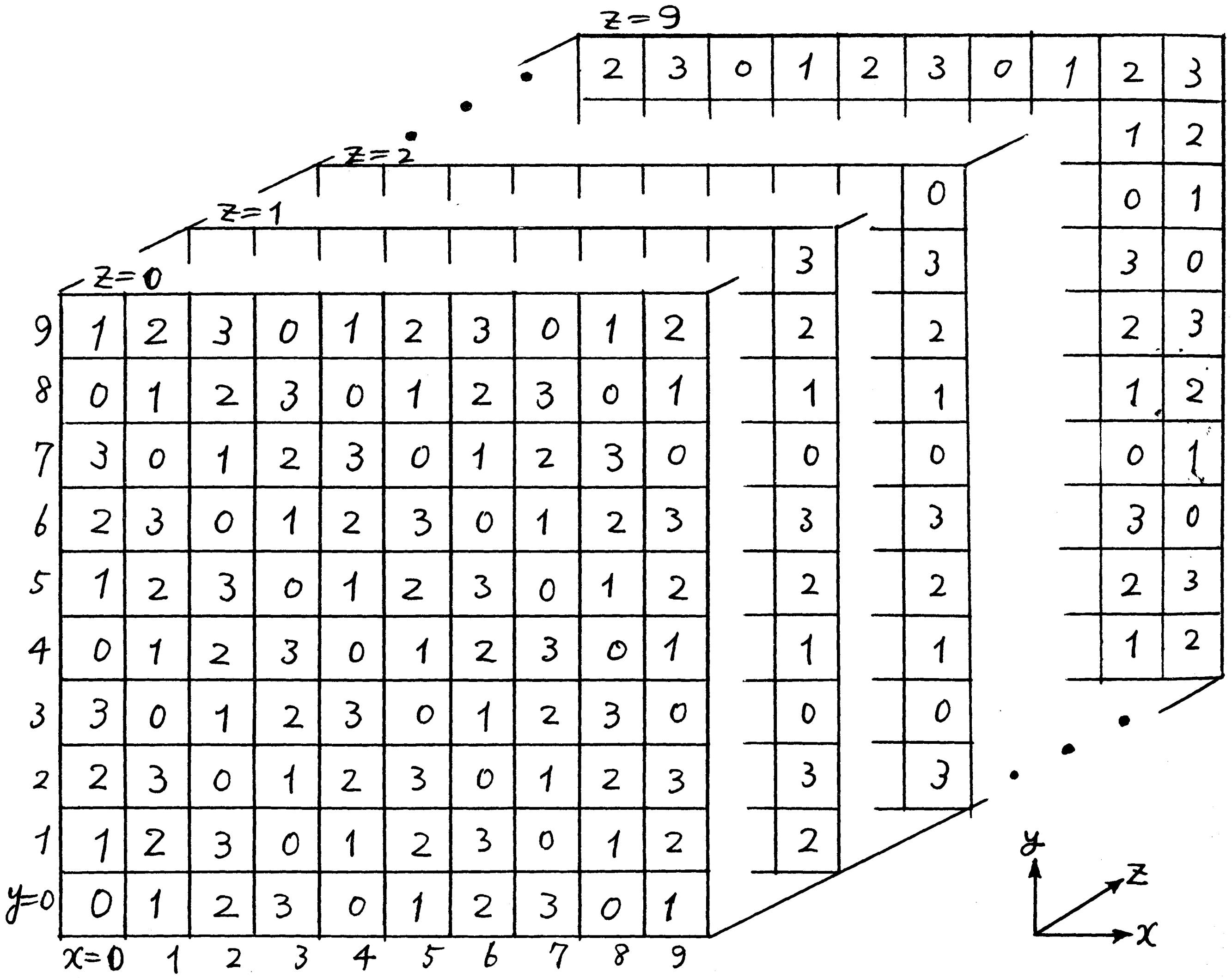
立体のパリティおよび一般化

同じ手法を立体の場合にも適用してみよう．問題は、10×10×10の立方体を、1×1×4の直方体250個を用いて組みたてることである．



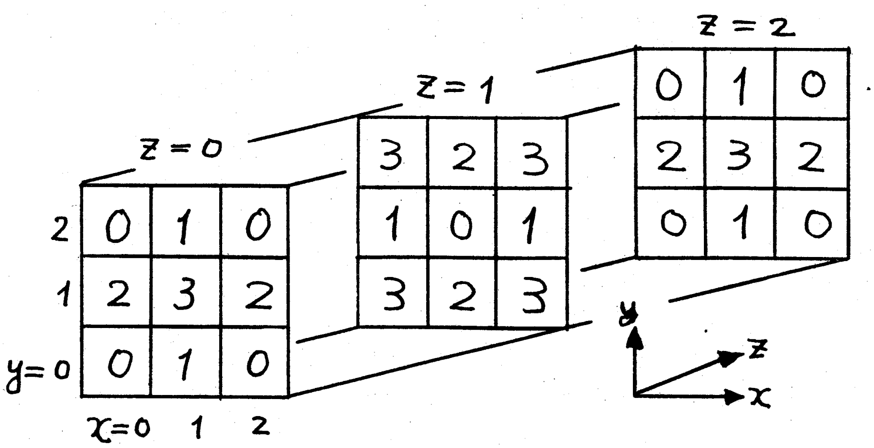
7	2	0	1	2	0	1	2	0
6	0	1	2	0	1	2	0	1
5	1	2	0	1	2	0	1	2
4	2	0	1	2	0	1	2	0
3	0	1	2	0	1	2	0	1
2	1	2	0	1	2	0	1	2
1	2	0	1	2	0	1	2	0
$y=0$	0	1	2	0	1	2	0	1
	$x=0$	1	2	3	4	5	6	7

平面図形の場合と同じ考え方で、0～3をぬりわけてみると、図のようになる。ここでは $x+y+z = 0 \sim 3 \ \underline{\mathrm{mod}}\ 4$ をぬりわけの規則にしている。$z=0$ の面をとると、0が25個、1が26個、2が25個、3が24個になっている。全体の数を算出すると、0と3が249個、2と3が251個であることがわかり、組みたてが不可能であることが証明される。
「可能である」ことの証明に用いられなくても、可能であ



る場合の条件を求めるために、同じ手法が使用できることがある．問題は 3×3×3 の立方体を、1×2×2 の直方体6個と、1×1×1 の立方体3個を用いて組みたてることである．（同じ問題をやや拡張したものとして、5×5×5 の立方体を 1×2×2 の直方体29個と 1×1×3 の直方体3個を用いて組みたてるものがある．また 1×2×2 の直方体29 個の代りに、いくつかをまとめて、1×2×4 の直方体13 個、1×2×2 の直方体1個、2×2×2 の立方体1個として、1×1×3 の直方体3個を加えた18片にしたものは、J. Conway のパズルとして知られている．原理は、3×3×3 の立方体とまったく同じである．）

1×2×4 の直方体をどのむきに置いても、0～3 が1個ずつおおわれるような，ぬりわけの方法の例を図に示す。各数を2進数で書いてみると、関係は明らかである．たとえば、



$\oplus$で排他論理和を表わせば、$\{(x \bmod 2) \oplus (z \bmod 2)\} + 2 \times \{(y \bmod 2) \oplus (z \bmod 2)\}$ によって、$(x, y, z) \to \{0, 1, 2, 3\}$ のぬりわけを与えると考えてもよい。0が9個、1〜3が6個ずつのパターンであることから、「1×1×1 の立方体は、0の位置のどこかに置く」という必要条件が求まる。これで探索の範囲がきわめて狭くなる。(実は、3×3×3 のどの断面 3×3 をとっても、奇数個であって、1×2×4 の直方体の断面は必ず偶数個であることから、1×1×1 の立体が必ずどの断面にも現われなければならない、という別の必要条件が、やはりパリティの性質から得られる。このことから、中央の位置 $x=y=z=1$ が 1×1×1 の立方体で占められることがわかって、原理的に一意な組みたてかたが求まるのである。)

プログラミングへの応用

このような組合せ論の証明手法を、どのようにプログラミング手法としてとりいれるべきかが、本稿で論じたい主題である。特に、人工知能用言語によるプログラミングで、この問題は興味がもたれている。

J.McCarthy のアプローチは、一階の述語論理による記述であり、Resolution 法のプログラムに入力して、証明を得ようというものである。McCarthy の記述例を、次ページに示す。

$$I8 = \{0,1,2,3,4,5,6,7\}$$

$$\mathrm{Board} = I8 \times I8 - \{(0,0),(7,7)\}$$

$$\mathrm{Dominoe} = \{0,1\}$$

$$\forall x.\, x \in \mathrm{Placings} \equiv x \in (\mathrm{Board}^{\mathrm{Dominoe}}) \wedge \mathrm{adjacent}(x(0), x(1))$$

ただし $x(y)$ は $\mathrm{apply}(x, y)$ を示す

$$\forall x \forall y.\, \mathrm{Board} \ni x \wedge \mathrm{Board} \ni y \supset$$

$$[\mathrm{adjacent}(x,y) \equiv \alpha(x) = \alpha(y) \wedge \mathrm{next}(\beta(x), \beta(y))$$
$$\vee\ \beta(x) = \beta(y) \wedge \mathrm{next}(\alpha(x), \alpha(y))\ ]$$

ただし αは第1要素をとりだしβは第2要素をとりだす.

$$\forall x \forall y.\, x \in I8 \wedge y \in I8 \supset [\mathrm{next}(x,y) \equiv (x = y') \vee (y = x')]$$

ただし x'はxの successor を表わす.

$$\forall w.\, w \in \mathrm{Coverings} \equiv w \in \mathrm{Placings}$$
$$\wedge [\forall x \forall y.\, (x \in w) \wedge (y \in w) \supset (x = y) \vee \{x(0), x(1)\} \cap \{y(0), y(1)\} = \phi]$$
$$\wedge [\forall x.\, x \in \mathrm{Board} \supset \exists y.\, y \in w \wedge (x = y(0) \vee x = y(1))]$$

定理 $\mathrm{Coverings} = \phi$ を示すことが問題を解くこと.

補題 $\forall w.\, w \in \mathrm{Coverings} \supset \mathrm{card}\left[\bigsqcup_{x \in w} (\{x(0), x(1)\} \cap \mathrm{Black})\right]$
$$= \mathrm{card}\left[\bigsqcup_{x \in w} (\{x(0), x(1)\} \cap \mathrm{White})\right]$$

$\mathrm{Card}[\mathrm{Black}] = 32$; $\mathrm{Card}[\mathrm{White}] = 30$.

$$\bigsqcup_{x \in w} [f(x) \cap A] = \left[\bigsqcup_{x \in w} f(x)\right] \cap A$$

・・・・・ (以下略)

明らかに、どこまでの情報を公理系として与えるかが、ここでの問題である。R.W. Floyd による inductive assertion 法を用いた、プログラムの正当性の証明でも、assertion の選択がキイポイントである。D.I. Good によるこの方法の(*)自動化でも、assertion の選択は人間がおこない、証明過程の一部が自動化されている。述語論理の場合には、公理系として与える情報が広すぎることはかまわない点、すなわち実際の証明には使用しない情報が含まれていても、(能率低下以外)害をおよぼさない点は有利である。

PLANNER、MICROPLANNER の系列の言語では、必要になるかもしれない性質を枚挙しておき、どの性質を利用すべきかをアドバイスの形で表わすことによって、述語論理の場合よりも柔軟性に富んだ形で、情報を表現できる。

Dijkstra 流の Structured Programming の手法では、プログラムの設計の際に、まず問題の対象に関するプログラマの知識を枚挙することからはじめて、top down にプログラムを設計してゆく。パリティの一般化によって、探索の可能性を狭くしぼるといった考え方は、この初期段階の知識の枚挙から得られるべきものであり、プログラムの stepwise refinement の過程で突然発見できる種類のものではない。

最近の傾向として、対話形式のプログラム操作システムの

形で、マン・マシン・インタフェースを整えようとする動きがある。プログラミング言語は単独でインプリメントするより、言語システムの形にインプリメントしようというものである。LISP では既に INTERLISP など、この形をとるものが少くないが、従来インタプリティヴではなかった言語についても、この考えかたをとろうとするものである。

このようなシステムでは、プログラマが言語システムを介して、試行錯誤をおこなうことができ、その結果をプログラムにフィードバックできる。パリティの色わけ方法についても、いくつかの方法を指定して、実際のぬりわけの状況の打ち出しや色の数の集計などは、コンピュータに処理させることができる。

プログラマが、パリティなどの性質を見抜くための補助手段として、たとえば数の表現を10進数に限らずに、任意のn進数を用いて表現できるようにしておくこと、既約分数表現や任意のベース・セットに関するresidue数表現もとれるようにしておくこと、などがあれば便利であろう。

プログラミングの過程と思考の過程を分離せずに、互にフィードバックしあうことと、コンピュータも試行錯誤の際の道具として利用することが、heuristics をプログラミングに生かす方法であると思われる。

おわりに

現在の最も高速なコンピュータをもってしても、多くの組合せ論の問題を解くためには、素朴なプログラミングでは、時間的に不十分である。組合せ論で使用される種々の証明手法を、プログラミングの手法の一部にとりこんで定着させることにより、いくぶんかの向上が得られるであろう。パリティ検査およびその一般化の手法が、その端緒を開く有力な候補ではないかと考えているのである。

文献

前半は、David Klarner: CS150 Lecture Notes. (Stanford Univ.) (1972) によるところが大きい。このノートは、N. G. de Bruijn と共著の著書として発行されるはずである。また本稿に関連の深い部分は "Brick-Packing Puzzles." JRM 6-2, 112-117 (1973) にまとめられている。

後半は、John McCarthy: CS226 Lecture Notes. (Stanford Univ.) ed. by M. Newey (1972) による。

[1975年8月17日]