決定性有限オートマトンを用いた覆面算の数え上げ

野崎 裕樹[1]　吉仲　亮[1]　篠原　歩[1]

概要：覆面算とは，本来数字で表される部分が記号として表された等式から，その等式を成立させるような文字と数字の対応を探し当てるパズルの一種である．覆面算を解析する手法として決定性有限オートマトン（DFA）を用いた手法が提案されている．本稿ではより効率的な DFA の構築方法，および DFA の遷移に追加の情報を保存することによる DFA の状態削減手法を提案する．また提案手法を用いて構築された DFA から，覆面算の数を桁数についての一般項で示すための手法を提案し，実際に 2 進数，3 進数の覆面算の数の一般項を提示する．

Numeration of alphametics using deterministic finite automaton

Yuki Nozaki[1]　Ryo Yoshinaka[1]　Ayumi Shinohara[1]

Abstract: Alphametic is a mathematical puzzle. Players are given a mathematical formula consisting of letters rather than numerals, and try to find the right substitution of numerals for letters that makes the formula hold true. A preceding study has proposed a method using DFAs for analyzing alphametics. In this paper, we propose a more efficient method to construct DFAs, and to reduce the number of DFA states. We also describe a method which derive a general term formula about the number of alphametic and show general term formulas for the binary and ternary number systems.

1. はじめに

覆面算 (alphametics または cryptarithm) とは，計算式の 0 ～ 9 の各数字がそれぞれ別の記号に置き換えられた形で与えられたとき，どの記号がどの数字に対応しているかを推測し元の完全な計算式を導き出すパズルである．図 1 は覆面算の一例とその解である．一般に覆面算は，任意の項数の四則演算の式の形で与えることができるが，本研究では 3 項の加算の計算式のみを考慮する．また通常，覆面算は少なくとも 1 つの解を持つが，本研究では解を持たない場合も考慮する．さらに，10 進数のみならず，記号を数字で置き換えたとき k 進数の数字として計算した際に解を持つかも考慮する．

	s	e	n	d
+	m	o	r	e
m	o	n	e	y

	9	5	6	7
+	1	0	8	5
1	0	6	5	2

図 1　覆面算の例と解

数字に対応する記号を変えることにより同じ計算式を導く複数の覆面算を与えることができる．そのような覆面算を等価な覆面算と定義し，1 つと数える．本研究の最終目標は，任意の桁数に対する解を持つ覆面算を k 進数ごとに数え上げ，数独プロジェクト [1] のように各覆面算の列挙・番号付けを行うことである．

先行研究 [2] では桁数 n の覆面算が与えられたとき，その覆面算が k 進法のもとで解を持つかの判定を決定性有限オートマトン (DFA) を用いて判定する手法が提案されている．この手法では，覆面算が満たすべき複数の条件に着目し，それぞれの条件を判定する DFA をそれぞれ個々に作ったうえで，それらの積を取ることによって，解を持つ覆面算を受理する DFA を構築する．しかしながら，この手法では k が大きくなるにつれ個々の DFA の状態数が増え，積をとる処理に時間がかかり実用的とは言えなくなってしまう問題があった．実際，$k \leq 3$ 程度までしかこの DFA を明示的に構築することができていない．

覆面算の数え上げの手法は先行研究 [3] で提案されてい

[1] 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University

る．この手法では，k-進数での解を持つ n 桁の覆面算を数え上げるため，まず加算が成立する計算式を列挙し，その数字を文字で置き換えることで覆面算を作成する．しかしながらこの手法では，計算式の桁数 n が増えた際に加算が成立する計算式の数も増加し，処理時間が指数的に増加するため実用的な範囲で求めることのできる桁数 n は上限が存在する．

そこで本研究では，解を持つ覆面算を判定する DFA のより効率的な構築方法，及び k 進数ごとに解を持つ覆面算の数を n の式 $F_k(n)$ として導出する手法を提案する．

提案する DFA の構築手法では，解を持つ覆面算が満たすべき複数の条件を判定するための情報を DFA の各状態で管理することにより解を持つ覆面算を受理する DFA を直接構築する．さらに，構築された DFA には対象構造が存在することに着目し，DFA に情報を追加し拡張することで，より少ない状態数で解を持つ覆面算を判定できるデータ構造を提案する．その結果，先行研究 [2] と比較し高速に DFA を構築することに成功した．また先行研究 [2] では構築できなかった $k = 4, 5, 6$ の DFA の構築に成功した．

覆面算の数え上げでは隣接行列を用いることによって，k 進数ごとに解を持つ覆面算の数を桁数 n の式 $F_k(n)$ として導出する手法を提案する．これにより n の大きさに依存せず任意の桁数の解を持つ覆面算の数え上げを可能にする．実際に $k \leq 3$ における一般項 $F_k(n)$ を導出することに成功した．

本論文は以下の構成である．2 章で覆面算を形式的に定義し，3 章で解を持つ覆面算を受理する DFA の構築方法を示し，実際に構築する．4 章で 3 章で構築された DFA の状態数を削減する方法について述べる．5 章で構築された DFA から覆面算の数の一般項の導出方法について述べる．6 章の実験において構築した DFA や一般項を示し，7 章でまとめを述べる．

2. 定義

2.1 覆面算

要素数 $k \in \mathbb{N}$ の順序付き文字の集合を Σ_k と表す，Σ_k の要素を a,b,c... で表し，a < b < c . . . とする．0 から $k-1$ までの数字の集合を $N_k = \{n \in \mathbb{N} \mid 0 \leq n \leq k-1\}$ と表す．任意の N_k^* の要素である数字列を，k 進数の数と解釈する．

定義 1 (k 進数-覆面算) 系列 $w_1, w_2, w_3 \in \Sigma_k^*$ に対して，w_i を項と呼び，(w_1, w_2, w_3) を k **進数-覆面算**と呼ぶ．

写像 $\pi : \Sigma_k \to N_k$ を拡張し，Σ_k^* の要素の文字列を数字列に置き換える準同型写像を $\hat{\pi} : \Sigma_k^* \to N_k^*$ と表す．例えば，$k = 2$, $\pi = \{\mathrm{a} \mapsto 1, \mathrm{b} \mapsto 0\}$ のとき，$\hat{\pi}(\mathrm{aba}) = 101$ である．

定義 2 (解を持つ覆面算) 次の条件を満たす，単射 $\pi : \Sigma_k \to N_k$ が存在するとき，覆面算は**解を持つ**という．

(1) $\hat{\pi}(w_1) + \hat{\pi}(w_2) = \hat{\pi}(w_3)$ が成立．

(2) 各 w_i の先頭文字 $w_{i,top}$ に対し，$\pi(w_{i,top}) \neq 0$.

また，1 つの覆面算に対して上記を満たす π が複数存在した時，その覆面算を**複数解を持つ覆面算**と呼び，π が一つしか存在しない時，その覆面算を**唯一解を持つ覆面算**と呼ぶ．

2.2 覆面算系列

解をもつ覆面算の集合を正規言語に変換するための操作を定義する．正規言語に変換することにより DFA での表現が可能となる．

一般に覆面算の各項 w_i の長さは異なるが，DFA で表現する際は各項の桁数が等しい必要がある．

定義 3 (等項長覆面算) w_3 の最上位桁に終端記号 $ をひとつ加えた文字列を w_3' とし，$|w_1'| = |w_2'| = |w_3'|$ となるように w_1, w_2 の上位桁に必要な数だけ $ を加えた文字列を w_1', w_2' とする．w_1', w_2', w_3' によって与えられる覆面算 (w_1', w_2', w_3') を**等項長覆面算**と呼ぶ．

図 2 に $ を追加した例を示す．

	b	e	e	r	
+		a	n	d	
s	o	d	a	s	

$	$	b	e	e	r
$	$	$	a	n	d
$	s	o	d	a	s

図 2 等項長覆面算の例

$w_{i,j}$ を覆面算の i 項の下位桁から j 番目の文字とする．

定義 4 (覆面算系列) 等項長覆面算 (w_1, w_2, w_3) に対して，系列 $w_{1,1}w_{2,1}w_{3,1}w_{1,2}w_{2,2}...w_{n,m}$ を考える．これは等項長覆面算を筆算の形にして下位桁から縦に文字を読み連結させた文字列を表現する．この系列を**覆面算系列**と呼ぶ．この覆面算系列は正規言語であることが先行研究 [2] によって証明されている．覆面算系列の例を図 3 に示す．

$	$	b	e	e	r
$	$	$	a	n	d
$	s	o	d	a	s

rdsenaeadb$o$$s$$$

図 3 覆面算系列の例

文字から文字への写像 $\gamma : \Sigma_k \to \Sigma_k$ を拡張し，文字列から文字列へ置き換える準同型写像を $\hat{\gamma} : \Sigma_k^* \to \Sigma_k^*$ と表す．

異なる二つの k 進数-覆面算系列 r, r' に対して，$r = \hat{\gamma}(r')$ となるような単射 γ が存在するとき，r, r' を**等価な覆面算系列**と呼ぶ．本研究では等価な覆面算を同一の覆面算として扱うため，覆面算系列の正規化を定義する．

定義 5 (正規化された覆面算系列) 覆面算系列を先頭から末尾まで読み進め新規の文字の出現順序が辞書順になる時，その覆面算系列は**正規化されている**という．

例として，覆面算系列 r =abbacb$ac は正規化されているが，$r'$ =acbacb$ac は b の出現の前に c が出現しているた

め正規化されていない．この r' を正規化すると abcabc$ab となる．

2.3 覆面 DFA

解を持つ覆面算系列を受理する DFA について述べる．

定義 6 (遷移文字) $\Delta_k = (\Sigma_k \cup \{\$\})^2\Sigma_k \cup \{\$\$\$\}$ の要素を**遷移文字**と呼ぶ．

遷移文字は，各項の j 桁目を連結した文字列 $w_{1,j}w_{2,j}w_{3,j}$ を表現する

定義 7 (覆面 DFA) 以下の入力と出力を持つ DFA を**覆面 DFA** と呼び，$M_k = (Q_k, \Delta_k, \delta_k, q_0, f_k, g_k)$ と表現する．この時，Q_k は状態集合，Δ_k は遷移文字の集合，δ_k は遷移関数，$q_0 \in Q_k$ は初期状態，$f_k \in Q_k$ を複数解を持つ覆面算系列が到達する受理状態，$g_k \in Q_k$ を唯一解を持つ覆面算系列が到達する受理状態とする．

- 入力：覆面算系列
- 出力：正規化された覆面算系列が解を持つ，唯一解をもつ，解をもたない

DFA は受理，非受理の 2 値判定であるが覆面 DFA では入力の覆面算系列が，複数解を持つ，唯一解をもつ，解をもたないの 3 値の判定を行う．覆面 DFA は 2 種類の受理状態を持ち，f_k に到達した系列を複数解をもつ系列，g_k に到達した系列は唯一解を持つ系列と判定する．入力がどちらの状態にも到達しなかった時，その覆面算系列は解を持たないと判定される．

3. 覆面 DFA の構築

本節では k 進数-覆面算系列を入力とし，解を持つまたは唯一解を持つ k 進数-覆面算系列を受理する 覆面 DFA $M_k = (Q_k, \Sigma_{M_k}, \delta_{M_k}, q_0, f_k, g_k)$ の構築方法について述べる．覆面 DFA は，$j-1$ 桁目まで正しい覆面算系列の接頭辞であることを前提に，j 桁目の遷移文字列を読んでなお正しい覆面算系列の接頭辞になりうるときのみ次の状態へ遷移させ，最後に $$$ を読み込んで覆面算系列を受理する．ここで，受理状態へ遷移する覆面算系列とは次の条件をみたすことであった．

(1) 覆面算に直したとき正しい形をしている．
(2) 正規化されている．
(3) 解・または唯一解を持つ．

各状態で上記の条件が暫定的に満たされていることを判定できる情報を保持し条件を満たす遷移と状態を作成することで，直接正しい覆面算系列を受理する DFA を構築する．

各状態は，以下の 3 つの情報 D, ℓ, P を保持する．

- $D = d_1d_2 \in \{0,1\}^2$ は，第 1 項，第 2 項にすでに $ が出現しているかどうかを表現する．i 項に$が出現している時 $d_i = 1$ となる．$w_{i,j-1} = \$$ かつ $w_{i,j} \neq \$$ のとき上記 (1) の条件に違反するためその判定に用いる．
- $\ell \leq k$ は，遷移文字列に使うことのできる文字の種類数を表す．これは上記 (2) の条件の判定に用いる．
- P は 4 つ組 (π, c, z_1, z_2) を要素として持つ空でない集合である．

$$\pi : \begin{cases} \Sigma_\ell \to N_k & \text{if } \ell < k \\ \Sigma_k \to N_k & \text{if } \ell = k \end{cases}$$

は可能な割当であり，$c \in \{0,1\}$ は π の下での繰り上がりの有無，z_i は第 i 項の直前の文字が 0 に割り当てられているか，を表現している．すなわち，

$$z_i = \begin{cases} 0 & \text{if } \pi(w_{i,j-1}) = 0 \\ 1 & \text{otherwise} \end{cases}$$

これらは (3) の条件の管理のために用いられる．特に z_1, z_2 は，最上位桁に 0 が許されていないことに対応する．

状態は第 3 項にすでに$が出現しているかを表す情報を持たない．第 3 項に$が出現する遷移文字は$$$のみであり，これは覆面算の終端を表す．以降の判定は行われないため，不要である．また，覆面算では第 1,2 項のみならず，第 3 項の最上位桁が 0 になることを禁止しているが，これに関する情報が不要であることは以下の議論によって確認できる．もし，第 3 項の最上位桁に対して 0 を割り当てる π を解とするならば，π は以下の計算式を成り立たせることになる．

$$\begin{array}{ccc} \$ & \alpha_1 & \dots \\ \$ & \alpha_2 & \dots \\ \hline \$ & 0 & \dots \end{array}$$

ここで，$\alpha_1 = \alpha_2 = 0$ すなわち， π は第 1,2 項の最上位桁に対して 0 を割り当てているため，$z_1 = z_2 = 0$ となり，このような π は解でないことがわかる．

以降，状態 q の持つ情報をそれぞれ，D_q, ℓ_q, P_q と表す．覆面 DFA の状態 q から遷移可能な文字は，各遷移文字と q の持つ情報を比較し，上記条件に違反しない全ての文字である．各状態の情報は，状態に遷移する一つ前の状態の情報と状態に到達するときに読んだ遷移文字によって定まる．状態 q から遷移文字 $x = x_1x_2x_3 \in \Delta_k$ を読む場合を考える．遷移可能な場合，遷移先の状態を q' とする．このとき，$D_{q'}, \ell_{q'}, P_{q'}$ は次のように定まる．

$D_{q'}$ は各項にすでに $ が出現していたか否かであった．$d_{i_q} = 1$ かつ $x_i \neq \$$ の時，解を持つ覆面算の条件に違反し遷移不可能である．遷移可能な時，$D_{q'}$ は以下のように定まる．

$$\begin{cases} d_{i_{q'}} = 1 \text{ if } x_i = \$ \\ d_{i_{q'}} = 0 \text{ otherwise} \end{cases}$$

$\ell_{q'}$ は遷移文字列に使用可能な文字の種類数を表す．q からの遷移で使用可能な文字は q からの遷移でも使用可能である．そこで $\ell_{q'} = \ell_q$ で初期化し，x_1 から順に $\ell_{q'}$

と x_i を比較して行く．$x_i \notin \Sigma_{\ell_{q'}+1}$ のとき，正規化された覆面系列ではなくなるため条件に違反し遷移不可能である．$x_i \in \Sigma_{\ell_{q'}+1} - \Sigma_{\ell_{q'}}$ の時，新しい文字の出現となるため $\ell_{q'} = \ell_{q'} + 1$ と更新し比較を続ける．

P_q は可能な割当 π，割り当てに応じた繰り上げの有無 c と直前の文字が 0 に割り当てられるかを表現する z_1, z_2 の 4 つ組みの情報の集合を管理する．p の持つそれぞれの要素を $\pi_p, c_p, z_{1_p}, z_{2_p}$ と表す．$p' \in P_{q'}$ とした時，$P_{q'}$ は p と x に対して以下の全ての条件を満たす $(\pi_{p'}, c_{p'}, z_{1_{p'}}, z_{2_{p'}})$ の集合となる．

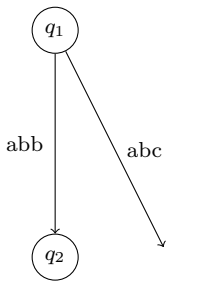
- $\pi p'(x_i) = \pi p(x_i) \ \text{ if } \ x_i \in \Sigma_{\ell_q}$
- $x_1 = x_2 = \$ \ \Rightarrow \ x_3 = \$ \vee p'(x_3) = 1$
- $p'(x_1) + p'(x_2) = p'(x_3) + c_{p'}k \ \ \text{ただし } p'(\$) = 0$
- $z_i = \begin{cases} 0 & \text{if } p'(x_i) = 0 \\ 1 & \text{otherwise} \end{cases}$

上記を満たす p' が存在しない時，条件に違反し遷移不可能である．

また \$\$\$ を読み遷移可能な場合，覆面算系列の終了を表すため受理状態へ遷移させる．このとき，可能な割り当て集合の数 $|P_{q_j}| > 1$ のとき複数解を持ち，$|P_q| = 1$ の時その覆面算系列は唯一解もつ．すなわち状態 q から\$\$\$を読んだ際の遷移先は以下のようになる．

$$\begin{cases} \delta_k(q, \$\$\$) = f_k & \text{if } |P_q| > 1 \\ \delta_k(q, \$\$\$) = g_k & \text{if } |P_q| = 1 \end{cases}$$

図 4 に $k = 3$ の場合の構築途中の覆面 DFA の例を示す．



各状態の持つ情報

- q_1
  - $D_{q_1} = 00$
  - $\theta_{q_1} = 2$
  - $P_{q_1} = \{(\{a \mapsto 0\}, 0, 0, 0)\}$
- q_2
  - $D_{q_1} = 00$
  - $\theta_{q_1} = 3$
  - $P_{q_1} = \{(\{a \mapsto 0, b \mapsto 1\}, 0, 0, 1), (\{a \mapsto 0, b \mapsto 2\}, 0, 0, 1)\}$

図 4 進数 3 の構築途中の DFA

状態 q_1 から abb で遷移した時，q_2 の情報は図 4 のように決定される．また，abc を読んで遷移しようとした時，可能な割り当てが存在しないため遷移不可能である．

状態 q から各遷移文字を読んだ遷移先は，すでに同じ情報をもつ状態が存在する場合その状態を遷移先とし，同じ情報を持つ状態が他にいない場合新しくその情報をもつ状態を作成し遷移先とする．

各状態で全ての遷移文字に対して，それぞれ遷移可能かを判定する．可能である場合次の遷移先の情報を作成し，情報に基づき遷移先を決定する．作成された全ての状態の遷移先を定めることにより，覆面 DFA が構築可能である．

Algorithm 1: 覆面 DFA の構築手法

```
Input: 進数 k
Output: 覆面 DFA M_k
M_k を初期状態 q_0 のみの DFA として初期化;
foreach 状態:q of M_k do
    foreach 遷移文字列:a of Σ_{M_k} do
        if a が正しい計算式の形式または正規化に違反 then
            continue;
        a,q から遷移先の D, θ を決定;
        foreach p of P_q do
            if p で加算が成立 or p に割当を追加し加算が成立 then
                遷移先の P に a,p で定まる (π, c, z_i) を追加 ;
        if 遷移先の P が存在しない then
            continue;
        if 状態 q' と遷移先の情報が一致 then
            δ(q, a) = q' ;
        else
            遷移先の情報を持つ状態 q'' を作成;
            δ(q, w) = q'' ;
output M_k;
```

図 5 に上記の方法で構築した 2 進数覆面 DFA を示す．$k = 2$ のとき，a に対応する数字が決まると，b に対応する数字も決まるため，常に唯一解を持つ．したがって，図 5 の受理状態は g_k の 1 つとなる．

4. 覆面 DFA の状態数の削減

前章の構築法を用いることにより唯一解を持つ，または解を持つ覆面算系列を受理する DFA が構築できる．このようにして構築された覆面 DFA は対称的な部分構造を複数箇所に持つ．

前章の手法で構築した 2 進数覆面 DFA を具体例とする．以降 q_i は図 5 の i でラベルづけされた状態をさす．初期状態から aaa,abb をそれぞれ読んで遷移する 2 状態 q_1, q_4 と，aab,aba それぞれ読んで遷移する 2 状態 q_2, q_3 に着目する．前者の 2 状態からともに到達できる状態の集合を Q，後者の 2 状態からともに到達できる状態の集合を Q' とした時，$|Q|, |Q'| > 1$ である．また，$|Q \wedge Q'| = 1$ であり，両者から遷移できる共通の状態は受理状態のみである．これは図 5 の DFA を見ても明らかである．初期状態から左右にそれぞれ独立した部分構造が作成され，最終的に一つの受理状態にまとまる構造をしている．$k = 2$ の場合，q_1, q_4 に遷移する文字は，$\{a \mapsto 0, b \mapsto 1\}$ の割り当てで加算が成立し，q_2, q_3 に遷移する文字は，$\{a \mapsto 1, b \mapsto 0\}$ の割り当てで加算が成立する．q_1, q_4 と q_2, q_3 の持つ割り当ての情報は異なるので，それぞれから作成される遷移先の状態の情報が

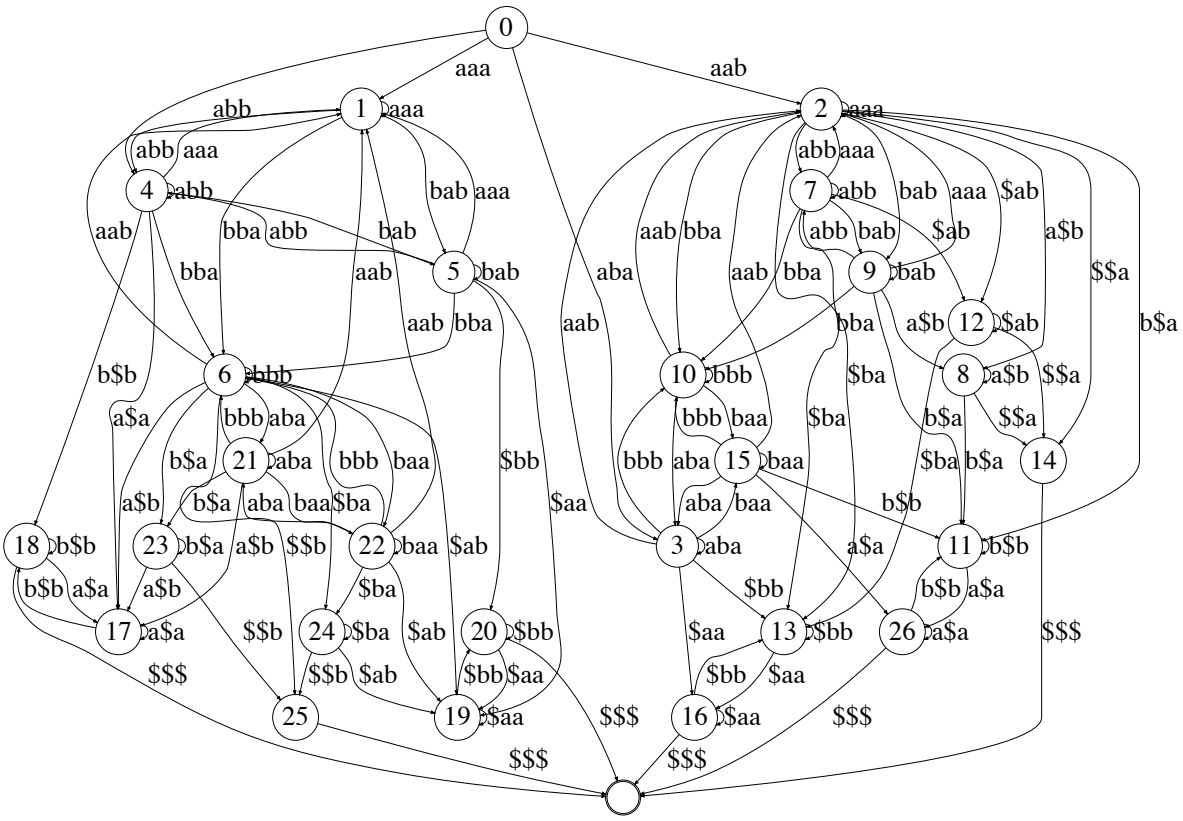


図 5 2 進数覆面 DFA

一致することはないため，独立した部分構造が作成される．

ここで，$q \in Q$ と $q' \in Q'$ の 2 状態が持つ情報を比較した時，割り当ての情報のみが異なる 2 状態が複数組存在する．図 5 の覆面 DFA での一例として，aaabab の系列を読み到達する状態 q_5 と abaaba を読んで到達する状態 q_3 の 2 状態に着目する．この時，$D_{q_5} = D_{q_3} = (0, 0)$，$\ell_{q_5} = \ell_{q_3} = b$，$P_{q_5} = \{(\{a \mapsto 0, b \mapsto 1\}, 0, (1, 0))\}, P_{q_5} = \{(\{a \mapsto 1, b \mapsto 0\}, 0, (1, 0))\}$ であり割り当ての情報のみが 2 状態で異なる．$\{a \mapsto b, b \mapsto a\}$ の写像を q_5 のもつ割り当てに適用した時 q_3 の情報と完全に一致する．このような割り当ての情報のみが異なる状態が複数組存在するため，対称構造が構築される．図 5 における割り当てのみの情報が異なる 2 状態 (q, q') を表 1 に示す．

表 1 割り当ての情報が同じ 2 状態

(1,10)	(4,15)	(5,3)	(6,2)	(17,11)
(18,26)	(19,13)	(20, 16)	(21,9)	(22, 7)
(23, 8)	(24,12)	(25,14)		

図 6 に $k = 2$ の場合の，上記の 2 状態のペアをまとめた DFA を示す．図に記載された写像は上記の 2 状態のペアをまとめることが可能な写像である．具体例として $k = 2$ の覆面 DFA について議論してきたが，$k \geq 3$ の覆面 DFA においても割り当ての情報のみが異なる状態の組が存在し，対象的な部分構造が作られる．k が増加するにしたがって割り当てが $k!$ で増加し，対称構造の数もそれに応じて増加する．

上記のような状態をまとめることにより，状態数の削減を行う．文字から文字への単射を $\gamma : \Sigma_k \to \Sigma_k$ とする．写像 γ は覆面 DFA の状態 q の情報 P_q が管理する複数の割り当てに関する情報の文字と数字の割り当てを γ にしたがって変更する．

$$\gamma(P_q) = \{p \in P_q \mid (\gamma \circ p_\pi, p_c, p_{z_1}, p_{z_2})\}$$

覆面 DFA 中の異なる 2 つの状態 q, q' において $D_q = D_{q'}, \ell_q = \ell_{q'}, P_q = \gamma(P_{q'})$ となるような γ が存在する 2 つの状態を**類似な状態**と呼び，この 2 状態がまとめる対象となる．q, q' をまとめる際，本来 q' に遷移をするはずの遷移文字列を q への遷移へと変更する．このとき可能な割り当てが変わってしまい，q から次に遷移文字列を読んで遷移する際に解をもつ覆面算系列の接頭辞でなくなったり，入力が解を持つ覆面算系列にもかかわらず受理されなくなったりしてしまう．これを防ぐために類似な 2 状態をまとめた際，q への遷移文字とともに写像 γ を遷移の情報として保持する．図 7 に $k = 2$ の場合の類似な 2 状態の 1 つの組をまとめる様子を示す．

状態 q_2 から baa の遷移先を作成する時，状態 q_4 が存在しない場合，前章の構築方法では新しい状態 q_4 が作成される．しかし，写像 $\gamma : \{a \mapsto b, b \mapsto a\}$ によって q_3, q_4 の

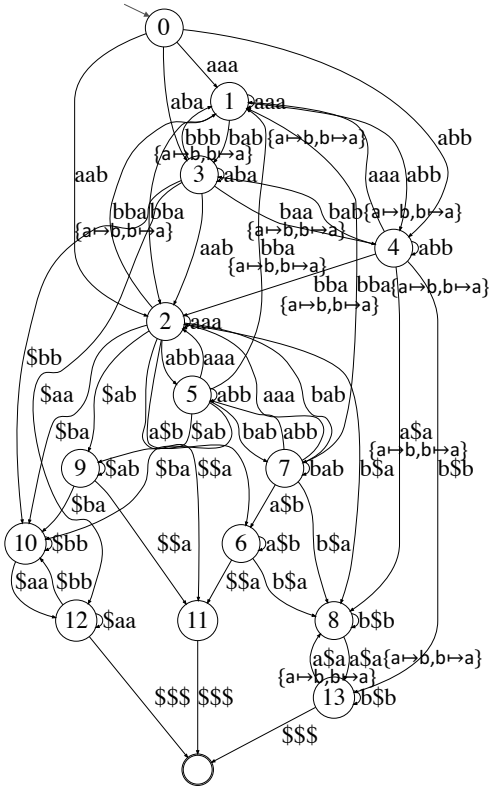
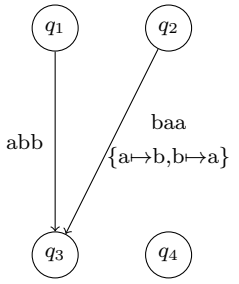


図 6 進数 2 の覆面 DFA



各状態の持つ情報

- q_1
  - $P_{q_1} = \{(\{\mathrm{a} \mapsto 0, \mathrm{b} \mapsto 1\}, 0, 0, 0)\}$
- q_2
  - $P_{q_2} = \{(\{\mathrm{a} \mapsto 1, \mathrm{b} \mapsto 0\}, 0, 1, 0)\}$
- q_3
  - $D_{q_3} = 00$
  - $\theta_{q_3} = 2$
  - $P_{q_3} = \{(\mathrm{a} \mapsto 0, \mathrm{b} \mapsto 1)\}, 0, 0, 1)\}$
- q_4(実際は作成されない)
  - $D_{q_4} = 00$
  - $\theta_{q_4} = 2$
  - $P_{q_4} = \{(\{\mathrm{a} \mapsto 1, \mathrm{b} \mapsto 0\}, 0, 0, 1)\}$

図 7 2 状態をまとめる

持つ情報が等しくなる．そこで，$\delta(q_2, baa) = q_3$ とし，γ を遷移とともに保持する．これにより状態 q_4 が新しく作成されず，状態数が削減される．

上記の類似な状態をまとめた覆面 DFA を**拡張覆面 DFA** と呼び，$EM_k = (Q_k, \Delta_k, \delta_{state}, \delta_{perm}, q_0, f_k, g_k)$ と表す．また，Π_k を Σ_k 上の順列の全体集合とする．このとき，Q_k は状態集合，Δ_k は遷移文字，$\delta_{state} : Q_k \times \Delta_k \to Q_k$ 状態遷移関数，$\delta_{perm} : Q_k \times \Delta_k \to \Pi_k$ は写像遷移関数，q_0 は初期状態，f_k は複数解を持つ覆面算系列が到達する受理状態，g_k は唯一解をもつ覆面算系列が到達する受理状態である．拡張覆面 DFA で遷移する時，状態の他に入力を変換する写像 $\gamma \in \Pi$ を保持する．$q \in Q_k, x \in \Delta_k$ とした時，$\delta : Q_k \times \Pi_k \times \Delta_k \to Q_k \times \Pi_k$ を次のように定義する．

$$\delta(q, \gamma, x) = (\delta_{state}(q, \gamma(x)), \delta_{perm}(q, \gamma(x) \circ \gamma)$$

ただし，$\gamma \circ \gamma'$ は合成写像を表す．$w \in \Delta_k^*, x \in \Delta_k$ とした時，拡張覆面 DFA の遷移は帰納的に表現される．

$$\delta^*(q, \gamma, \epsilon) = (q, \gamma)$$
$$\delta^*(q, \gamma, wx) = \delta(q', \gamma', x) \text{ ただし } (q', \gamma') = \delta^*(q, \gamma, w)$$

拡張覆面 DFA の入力 $r \in \Delta_k^*$ が唯一解を持つ場合と複数解を持つ場合で以下のように遷移する．

$\delta^*(q_0, \iota, r) = f_k$ if r が複数解を持つ覆面算系列

$\delta^*(q_0, \iota, r) = g_k$ if r が唯一解を持つ覆面算系列

ただし ι は $\forall v \in \Sigma_k, v = \iota(v)$ となる写像である．

実際の拡張覆面 DFA の構築では，前章の構築の過程で遷移先の状態の情報が定まった時，すでに作られた状態の中に類似な状態があるかを走査する．この時，類似な状態が見つからない場合は新しく状態を作成しその状態に遷移させる．類似な状態が見つかった場合は類似な状態に遷移を作り，遷移文字とともに写像を保存する．その結果，前章の構築方法では複数作成される類似な状態が全て 1 つの状態で表現されるため状態数が削減される．

5. 覆面算の数の一般項の導出

覆面 DFA は以下のような性質を持つ．

- 覆面 DFA は正しい覆面算を受理する．
- n 回の遷移で受理される覆面算系列は $n-1$ 桁の覆面算となる．（受理される覆面算系列は \$\$\$ を遷移文字列として遷移し受理されるため．）

よって，n 回の遷移で受理状態に至るパスの総数を数えることにより $n-1$ 桁の覆面算の総数 $F_k(n)$ を導出できる．

$F_k(n)$ の導出は覆面 DFA の隣接行列の累乗をもとめ導出する．覆面 DFA の隣接行列を A とする．A の各要素 $a_{i,j}$ は覆面 DFA の状態 q_j から q_i への遷移可能な遷移文字の数を表す．このとき，$A^n_{i,j}$ が状態 q_j から q_i へ n 回の遷移で到達するパスの総数となる．

開始状態から受理状態までのパスの数は，A^n の受理状態に対応する要素を取り出す行列 L を左から A^n にかけ，初期状態に対応する要素を取り出す行列 R を右から A^n にかけることによって求まる．

L は $1\times m$ の大きさの横ベクトルであり，R は $m\times 1$ の縦ベクトルとなる．L は解を持つ覆面算の数の一般項を求める場合と，唯一解を持つ覆面算の数の一般項を求める場合で異なる．解を持つ覆面算の一般項を導出する際は，L の列のうちと覆面 DFA の f_k に対応する列と，g_k に対応する列の二箇所を 1 とすることで導出が可能である．唯一解を持つ覆面算の一般項を導出する際は，L の列のうち f_k に対応する列を 1 とすることで導出が可能である．唯一解を持つ覆面算の一般項を求める時，L, R は以下のようになる．

$$L_{1,i} = \begin{cases} 1 & \text{if } (q_i\text{が唯一解を受理する状態のとき}) \\ 0 & \text{otherwise} \end{cases}$$

$$R_{i,1} = \begin{cases} 1 & \text{if } (i = q) \\ 0 & \text{otherwise} \end{cases}$$

一般項 $F_k(n)$ は A^n の各要素 $a_{i,j}$ を n の式で表すことができた時，求めることが可能である．

具体的に，図 8 に示す簡単なオートマトン M_{ex} を用いて，受理状態に n 回の遷移で到達できるパスの総数を求める．この DFA の隣接行列から導くことができる $A^n_{M_{ex}}$ を図 9 に示す．

一般項を求める時，状態 q_1 が開始状態，q_6 が受理状態のため，一般項 $F_{M_{ex}}(n)$ は以下になる．

$$F_{M_{ex}}(k) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} A^n_{M_{ex}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$F_{M_{ex}}(n) = 2 + (-1)^{n+1}$$

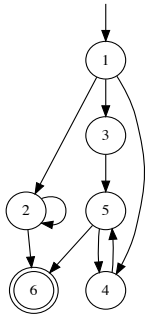


図 8 具体例 M_{ex}

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1+(-1)^{1+n} & 0 & \frac{1}{2}(1+(-1)^n) & \frac{1}{2}(1+(-1)^n) & \frac{1}{2}\left(1+(-1)^{1+n}\right) & 0 \\ 1+(-1)^n & 0 & \frac{1}{2}\left(1+(-1)^{1+n}\right) & \frac{1}{2}\left(1+(-1)^{1+n}\right) & \frac{1}{2}(1+(-1)^n) & 0 \\ 2+(-1)^{1+n} & 1 & \frac{1}{2}(1+(-1)^n) & \frac{1}{2}(1+(-1)^n) & \frac{1}{2}\left(1+(-1)^{1+n}\right) & 0 \end{pmatrix}$$

図 9 $A^n_{M_{ex}}$

表 2 覆面 DFA の状態数

進数 k	2	3	4	5	6
先行研究 [2]	243	1180	307233	-	-
提案手法 (まとめない)	28	110	859	10267	370719
提案手法 (まとめる)	15	27	163	1062	19252

表 3 解を持つ覆面算の数の一般項 $F_k(n)$

進数 k	$F_k(n)$
2	$3\times 2^{n-2}(2^{n-1}-1)$
3	$-3^{n-1}-2\times 5^{n-1}+4\times 9^{n-1}$

表 4 唯一解を持つ覆面算の数の一般項 $F_k(n)$

進数 k	$F_k(n)$
2	$3\times 2^{n-2}(2^{n-1}-1)$
3	$3^{n-1}-4\times 5^{n-1}+4\times 9^{n-1}$

6. 実験

実際に，$2\le k\le 6$ の範囲で覆面 DFA を構築し，$k=2$ の場合の覆面 DFA を図 6 に示す．$k\ge 3$ の覆面 DFA に関しては状態数が多いため省略する．

先行研究 [2] で構築された DFA，提案手法のうち類似な状態をまとめていない覆面 DFA，類似な状態をまとめた覆面 DFA それぞれの状態数を表 2 に示す．先行研究 [2] では，$k\ge 5$ のとき構築時間が長く構築できなかったため状態数の表記が $-$ となっている．提案手法では条件を判定するための情報を各状態で持つことにより直接 DFA を構築しているため先行研究 [2] と比較し高速に覆面 DFA の構築が可能である．そのため，$k=5,6$ の覆面 DFA の構築を可能とした．また，類似な状態をまとめることにより状態数の削減が行えたことも確認できた．しかしながら，

表 5 解をもつ覆面算の総数

先行研究

基数＼桁数	1	2	3	4	5	6	7
2	0	3	18	84	360	1488	6048
3	1	23	265	2639	24913	229703	2093785
4	2	69	1463	26716	456639	7561377	123194460
5	2	115	4622	148483	4184478	110899540	-
6	2	123	8650	498307	22931188	-	-
7	2	129	11108	1132397	-	-	-
8	2	124	11768	1701945	-	-	-
9	2	129	11831	1882449	-	-	-
10	2	123	11935	1935453	-	-	-

提案手法

基数＼桁数	1	2	3	4	5	6	7	8	9	10
2	0	3	18	84	360	1488	6048	24384	97920	329448
3	1	23	265	2639	24913	229703	2093785	18973439	171399073	1545756023
4	2	69	1463	26716	456639	7561377	123194460	1990281467	32011044231	513628524308
5	2	115	4622	148483	4184478	110899540	2852251360	72299094358	1819642046811	45638896933615

$k \geq 7$ の時，状態数が多くなってしまい類似な状態の探索や遷移先を決める状態の数が増加してしまい，構築することができなかった．

$k = 2, 3$ の場合の解を持つ覆面算の数の一般項 $F_k(n)$ を表 3 に示し，唯一解を持つ覆面算の数の一般項 $F_k(n)$ を表 4 に示す．本実験では，数式処理ソフト Mathematica [4] を用いることによって一般項を求めた．

$k = 2$ の場合は，a の割り当てが決まった時に b の割り当ても一意に定まるため，全ての覆面算が唯一解をもつ覆面算となる．

$k \geq 4$ の時，Mathematica [4] を用いて一般項を求めることは不可能であった．これは k の増加とともに覆面 DFA の状態数が増加し，隣接行列の大きさが大きくなることや，連立漸化式の式の数が増え，一般項を求める処理に時間がかかるためである．

そこで先行研究 [3] で行われていたように，長さ n, 進数 k ごとに覆面算の数を求め，その結果を表 5 に示す．先行研究 [3] の手法では，$k = 10$ においても $n = 4$ までの数え上げを行うことはできているが，n の増加とともに正しい計算式から覆面算への変換の処理時間が増加するため数え上げが困難になっている．提案手法では $k \leq 6$ までの DFA しか構築できていないため $k \geq 7$ の覆面算の数え上げは不可能である．また，$k = 6$ に関しては状態数が増えるため隣接行列の累乗を求めるのに時間がかかり数え上げが困難であった．しかしながら，$k \leq 5$ の覆面 DFA の状態数が比較的少ないため，隣接行列の積を求めることが可能である．n 桁の覆面算の数は隣接行列の n 乗で求めることができる．そのため，先行研究 [3] では不可能である $F_k(100)$ のような大きい桁数の覆面算の数え上げが可能である．

7. まとめ

本論文では解を持つ覆面算系列，または唯一解を持つ覆面算系列を受理する DFA の構築方法と，覆面 DFA 中の対称構造に着目した状態数の削減をする手法を示した．また，その DFA から解を持つ，または唯一解を持つ覆面算系列の数に関する一般項を求める解析手法を提案し実際に解を持つ覆面算と唯一解を持つ覆面算の数の一般項を進数ごとに導出した．先行研究 [2] と比較し，その構築方法を変えることで状態数の削減を可能にし，先行研究 [2] では構築不可能であった進数の覆面算系列を受理する覆面 DFA の構築を可能とした．また，先行研究 [3] では，覆面算の数を長さ及び進数ごとに求めていたが本研究では長さに関する一般項を進数ごとに求めることを可能にした．

参考文献

[1] 井上真大, 奥乃博. 本質的に異なる数独解盤面の列挙と番号付け. 情報処理学会第 71 回全国大会, Vol. 5, p. 6, 2009.

[2] 遠藤洋, 成澤和志, 篠原歩. 覆面算を解析するためのオートマトン理論的アプローチ. ゲームプログラミングワークショップ 2011 論文集, Vol. 2011, No. 6, pp. 54–61, 2011.

[3] 遠藤洋. 有限オートマトンとその拡張モデルを用いたパズルの解析に関する研究. Master's thesis, 東北大学大学院情報科学研究科, 2013.

[4] Wolfram Research, Inc. Mathematica, Version 11.2. Champaign, IL, 2017.