組み込みソフトへの数理的アプローチ

第4回 論理式の実験場を作る，全数チェックへの道
―― 数理的アプローチの基礎トレーニング 2

藤倉 俊幸

1 前回のおさらい

前回（2006年11月号）は，論理式の意味を理解するために，与えられた論理式を真にする場合と偽にする場合を実際に作り出してテストする方法を説明した．しかし，

$$\forall x \exists y (Q(x, y) \rightarrow Q(y, x)) \cdots\cdots (1)$$

を偽にする場合を作り出せなかった．この論理式がつねに真であることは，タブロー法などでだれでも証明できるし，プログラム言語Prologで作ったプログラムなどでも証明できる．

ところが，$Q(x, y)$という述語を，「xはyが好き」と考えたとき，この論理式は，「すべての人(x)がそれぞれだれか(y)を愛するならば，そのだれか(y)からも愛される」という意味になり，これがつねに真であれば世の中から片思いがなくなってしまう．こんなことは，にわかには信じられない．証明されても納得がいかないのでどうしようか――というのが今回のテーマである．

疑問点は，なぜつねに真になるのかということと，本当に片思いはなくなるのか，の二つである．

2 実験場の開発と反例探し

● 偽の場合を探す

まず，証明を疑ってみよう．つまり，本当は偽になる場合があるかもしれない．

前回はVDM++を使用していろいろな場合で試してみたが，すべての場合を確認したわけではないし，もしかしたら見落としているかもしれない．反例が一つでもあれば証明の結果を覆すことができるが，すでにいろいろな場合をテスト済みなので，残された道は力業（ちからわざ）の全数検査しかない．

$Q(x, y)$の全数検査といっても，そんな途方もないことはできないので人数を制限して，3人しかいない世界を考えることにする．

3人いるときの関係をどのように表現すれば良いか考えてみる．グラフで描くと，**図1**のようにノードで人を表して矢印で関係を表すことができる．3人いるときのすべての関係はノードを3個に固定して，すべての矢印の引き方を網羅すれば良いことになる．どんな矢印の引き方があるかは，**図1**のマトリックス型を見れば一目瞭然で，全部で9通りあることがわかる．この9通りをリストすると，**図2**の左側のようになる．そして，この9通りを適当に組み合わせれば，一つの世界ができあがる．たとえば，**図1**の例は，次の三本の矢印でできた世界である．

{a→b，b→c，c→a}

このような組み合わせのすべてを作れば，3人の世界のすべてを作ることができる．組み合わせの作り方は，9本の矢印の集合の部分集合が一つの世界になるので，すべての部分集合を作ればよい．つまりベキ集合を作れば，すべての世界の集合を生成したことになる．**リスト1**にすべての世界を生成するためのVDM++モデルを示す．

リスト2に$\forall x \exists y (Q(x, y) \rightarrow Q(y, x))$の全数検査プログラムを示す．この関数`alltest22`が，trueだけの配列を返せば，この論理式P22がつねに真であることを疑う余地はなくなる．

リスト1と**リスト2**を見ただけでは，モデルの全体はわからないと思う．これらの関数は，前回作成した論理モデル・クラス（前回の図12）の操作として追加することで使用できる．あるいは，論理モデル・クラスのサブクラスとして実装すれば良い．実行結果の一部を**図3**に示す．

実験の結果はすべてtrueになった．しかし，残念ながら3人の世界までしか確認できなかった．4人の場合は2^{16}個の世界を検査するのでVDMがやってくれなかった．結局，3人までしか調べられなかったが，反例が出なかったことから証明を信用するしかないようである．

● プログラムの改良

プログラムをくふうすれば，4人の場合でも証明できる可能性があるが，3人の世界で計算時間が30秒ほどかかったのと，4人の次は5人ときりがないので，改良の方針のみを考えることにする．プログラムを改良するとすれば，ベキ集合は整数型の各ビットを集合要素に対応させて1のビットに対応した要素だけをピックアップすれば，すべての部分集合を作ることができる（前回のコラムを参照）．したがって，allRelation(k)の中でベキ集合を生成せず，部分集合を作りながら論理式の評価を行えば良さそうである．いわゆるon-the-fly型である．それで

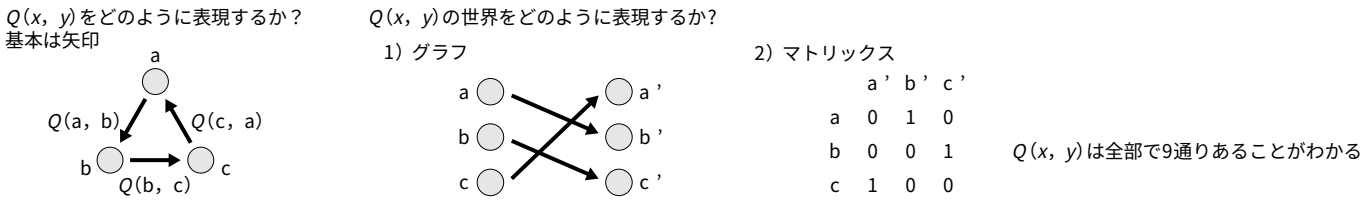

図1　関係をどのように表すか

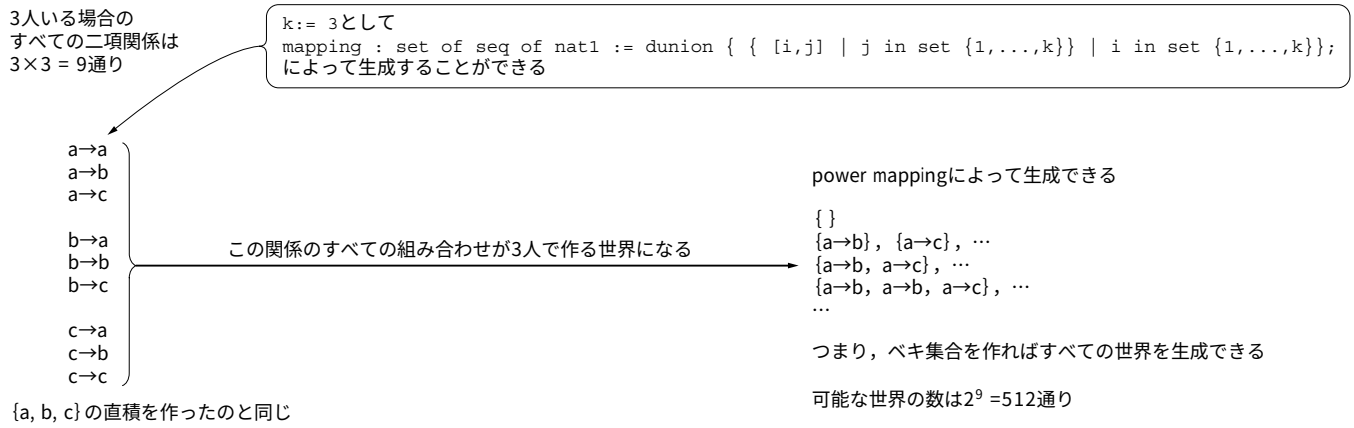

図2　すべての関係を作り出す

リスト1　世界を生成するプログラム

```
public allRelation : nat1 ==> set of seq of set of nat1
allRelation(k) ==                    世界における人数をkで指定する
    (
    dcl rlst : set of seq of set of nat1 := {};
    dcl lst : seq of set of nat1;
    dcl mapping : set of seq of nat1 := dunion { {
     [i,j] | j in set {1,...,k}} | i in set {1,...,k}};
    for all m in set power mapping do
        (lst := [ { } | i in set {1,...,k} & i > 0 ];
        for all a in set m do
            lst(a(1)) := lst(a(1)) union {a(2)};
        rlst := rlst union {lst});
    return rlst
    );
```

リスト2　Q(x, y)全数テスト

```
-- ∀x∃y(Q(x,y) => Q(y,x))
public P22 : () ==> bool
P22() == return forall x in set elems model &
 (exists y in set elems model & (x.Q(y) => y.Q(x)) );


public alltest22 : nat1 ==> seq of bool
alltest22(k) ==
    (
    dcl ans : seq of bool := [];
    setM(k);
    for all lst in set allRelation(k) do
        (setRelation(lst);
        ans := ans ^ [P22()];
        );
    return ans;
    );
```

も，int のビット長が制約になるし，処理が複雑になるので，連載の主旨からいえばそのプログラムが正しいことを証明しなければならない．そうすると話が長くなるので，3 人の世界だけでがまんすることにする．数が多くなってしまうのは，モデル検査などの全数検査を行う手法の宿命である．

● 夏目漱石の例

コンピュータを使っていると一見，頭を使っているようだが，実は力業に頼って安直な方向に走っているのではないだろうか．夏目漱石の場合は，3 人の世界を**図 4** に示した六つの世界にまとめている．これで，「こころ」とか「三四郎」で利用する人間関係を推こうしていたのかもしれない．実は，a，b，c を区別せずに純粋に関係だけに着目して分類するとこうなる．グラフの同型という性質を利用して分類するのである．今考えている論理式のテストでは，a，b，c を区別する必要はない．そのため，

```
>> create m1:= new QxyTest()          論理モデル・クラスの
>> print m1.alltest22(2)              サブクラスにした
[ true,true,true, …,true ]
                                      ここはすべてtrueになった．
>> print m1.alltest22(3)              ただし，3のときは30秒程度
[ true,true,true, …,true ]            の計算時間がかかる

>> print m1.alltest22(4)
C:/…/VDM/論理モデル.vpp, l. 38, c. 25:
  Run-Time Error 79: Set too big for 'power'
                                                    - limit is 16
4人の世界は，要素数が4×4=16個のベキ集合を作れず，実験
できなかった．limit is 16だが，16はだめなのか?
```

図3　実行結果の一部