論理プログラミングによる多数決投票ルールの分析

An analysis of simple majority rule by logic programming

犬童健良

2006.7.30

概要

本論文では，社会的選択問題を解く PROLOG プログラムを紹介した．またこれを用い，多数決投票が推移性や戦略的操作不可能性を満たす場合について，シミュレーション実験を行った．

1. はじめに

社会的選択理論は，投票や市場経済などの社会的意思決定のメカニズムを分析するための基礎であり，経済学，政策科学，政治学，倫理学，社会学といった分野で伝統的に応用されてきた(Sen(1982)；Arrow *et al.* (2002)；奥野ら(1988))．しかし，むしろ離散数学ないし組合せ最適化に近いそのモデリングは，論理学の言語に自然になじむ．

それゆえ，論理プログラミング言語 PROLOG(Clocksin and Mellish, 2003)は，個人や社会の持つランキング（選好関係）やその論理的な制約条件を表現するのに適していると考えられる．また PROLOG システムは自動定理証明マシンとして動作するので，論理的条件を適切にコード化すれば，後は導出原理とバックトラック機構により，所与の条件を満たす意思決定ルールや選好モデルを，メモリと時間が許す限り，調べ上げてくれる．

本論文では，PROLOG を用いたシミュレーション実験を通じて，社会的選択問題にかんする，以下の基礎的な問題を，3 代替案，3 人の社会について，解いてみたい．

- 多数決ルールを PROLOG のコードによって実装すること．
- 多数決ルールが推移性を満たす許容的領域を，実験的に生成すること．
- 推移性を満たす多数決ルールが，操作不可能であるかどうか調べること．
- 同様に，非独裁的であるか，市民主権性を満たすかどうかについても調べること．

PROLOG 言語の特質として，たんにプログラミング言語として，コード化ないし実装に用いられるだけではなく，それ自体が，論理的な意味を表すものである（主張１）．また実装されたコードは，PROLOG システムの定理自動証明メカニズム（導出原理）により，実行可能である（主張２）．それによって論理的モデルの正しさを実験的に検証（証明）することができるわけであるから，数学的言語による定式化と証明の組合せに代替しうる（主張３）．

本論文の以降の部分では，以上の 3 つの主張を背景に，4 つの問題を順に解いていくことにする．その前にまず社会選択理論について，簡単に紹介しておこう．

2. 社会的選択問題とは

社会的選択問題とは，社会状態（共通の代替案）の集合上の，人々の優先順序（ランキング）ないしペア比較を，しかじかの条件を満たすように，社会全体で集約することである．

しかじかの条件と言うのは，集約の結果が合理的であること（社会的ランキング，あるいはペア比較の推移性），独裁者が発生しないこと（非独裁性），虚偽申告によって得をする者が現れたりしないこと（戦略的操作の不可能性），あるいは特定の代替案が強制されたりしないこと（市民主権性）などのことである．

現代の社会的選択理論が確立したのは，上記の望ましい条件を満たすことが，常識的に考えられるよりはるかに難しいことを示す一連の定理が相次いで証明された，20 世紀中葉である (Arrow, 1951/1963; Gibbard, 1973; Satterthwaite, 1975)．18 世紀に，Condorcet と Borda が多数決投票のパラドックスや，順位評定法の戦略的操作について論争したことは，その先駆けといえる．また Arrow の定理以降，投票パラドックスの発生確率や，さまざまな選挙方式についての実験研究がさかんに行われた（入門書として，奥野ら(1988)，佐伯(1980)，宇佐美(2000) などを参照されたい)．最近になって，犬童(2006)と Indo (2006) は，PROLOG を用いて，Arrow の定理や Gibbard-Satterthwaite 定理の 2 人 3 代替案の場合を証明し，また操作不可能性と非独裁性を満たす社会的選択ルール (SCF) を持つ許容的領域を調べ上げている．

投票のパラドックスは，多数決ルールが推移性を満たさないとき起きる．May(1952)やMurakami(1966)は，多数決投票のパラドックスと Arrow の定理の関係を，3 値論理学を用いて探求している．多数決の推移性が保証されるための，したがって Arrow 流の社会的厚生関数（多数決 SWF）となるための必要十分条件は，Kenichi Inada(1969)によって解かれた．とくに強順序と人数の奇数性を仮定すると，価値制限(value restriction)と呼ばれる条件に集約される（Inada, 1969; Sen, 1982).

（定義）価値制限．任意の 3 代替案について，以下の 3 つの条件のいずれかに該当する場合.

- 最悪でないことに皆が合意する案がある（単峰性）.
- 最善でないことに皆が合意する案がある（単谷性）.
- 最悪でも最善でもないことに皆が合意する案がある（2 分割可能性）.

価値制限は，Black や Arrow の研究した単峰性(single peakedness)を一般化したものであり，その意味で投票者のランキングがお互いに類似していることを要請する．また許容的領域がラテン方格（図 1 のランキング番号でいうと，145 と 236 の 2 つの 3 つ組み）を含まないことに等しい．[1]

さらに，社会的意思決定関数（多数決 SDF）の場合，すなわち推移性に固執せず，すべて無差別の場合も含めて，ペアワイズ多数決で他の代替案すべてに対して劣らぬ案，すなわちコンドルセ勝者が選べさえすればよいと考えるならば，上記の条件から奇数性を落とせることが知られている(Sen and Pattanaik, 1969; Sen, 1982).

3. 許容的領域，多数決投票，および価値制限の論理プログラミング

本節では，PROLOG によって，3 人 3 代替案の場合の許容的領域，多数決ルール，および価値制限の条件を論理的にモデリングする．次節で，これらをシミュレーションで実験的に作り出

[1] Sen(1966)は価値制限を最初に導入し，多数決が推移的であるための十分条件となることを証明した．一方，「ラテン方格なし」の条件は，Ward(1965)によって十分性が示されたが，Sen はその強順序における同値性を指摘した(p.496)．弱順序の場合やその後の研究については，文献を参照せよ(Sen, 1982; Arrow et al., 2002; Gaertner, 2001).

し，その性質を確かめてみよう．

3 代替案で強順序（線形順序）の許容的領域Ωは，全部で 2 の 6 乗（2^6＝64）通りあるが，厳密には非制限的領域Σと空領域[]は除かれる．ただし，比較のため，これらについても取り除かずに実験することにする．なおこの部分の PROLOG コードは，以前の研究(犬童，2006; Indo，2006)で 2 人 3 代替案の場合について作られていたものを拡張した．

3 代替案の場合であれば，6 通りのランキングが可能であり，これをΣ＝｛r（J）：J=1,2,3,4,5,6｝と書く．図 1 では 6 つの事実節 possible_rankings_0/2 に対応する．スラッシュ後の数字は，述語の引数の数（アリティ）である．許容リスト＝許容的領域は，その一部を落としたものであり，Ω⊆Σとして表せる．最も緩やかなランキングの許容リストは，すべての可能なランキングを含む（非制限的領域と呼ばれる）．

図 1 では単一引数の事実節 admissible_rankings /1 が，引数として許容リストを格納する．これはプログラム読み込み時に，指令：- dynamic により動的変更を宣言されており，auto_restricted_domain/1 というルール節によって，設定される（コードは省略する）．possible_rankings/2 は現在の許容的領域におけるランキングである．

次に，図 2 に多数決ルール，および図 3 の左右に価値制限のコードを，2 通り示した．さらに図 4 にラテン方格を用いた価値制限の代替的なコードをそれぞれ示す．

図 2 のコードでは，各プロフィール（3 人のランキングの組）において，PROLOG のルール節 is_simple_majority_decision/2 を用い，各ペア比較を，3 人の単純多数決投票によって決めている．また多数決に基づく社会的選択ルール（SCF)を自動生成する PROLOG のルール節 auto_scf/2 では，プロフィールについての再帰により，is_Condorcet_winner/2 を用いて，それぞれ一つの代替案（コンドルセ勝者）を割り当てている．

したがって図 2 のコードは，図 1 のコードが生成する許容的領域に対して，多数決に基づく社会的選択ルール（SCF）をもれなく生成することができる．

4. 多数決投票のシミュレーション実験

本節では，図 1 と図 2 に示した PROLOG プログラムを使って，シミュレーション実験を行う．まず，図 5 に，3 種類の価値制限のいずれかを満たす許容的領域を生成した結果を示す．[2]

Inada の定理により，奇数人数のときは，価値制限領域において，多数決ルールがつねに推移性を満たし（すなわち多数決 SWF になり），またこれらの領域以外では，多数決ルールが推移性に違反する場合が生じる(Inada, 1969)．

上記の必要十分性は，図 2 のコードを用いた実験によって，簡単に確かめることができる．多数決ルールを表す is_simple_majority_decision/1 は， is_Condorcet_winner/2 によって投票結果を集計するが，代替案ペアに対する多数決による決定（すなわちペアワイズ多数決）が強順序で循環し，非推移性が生じる（すなわちコンドルセ勝者がいなくなる）場合がある．またそれらの非推移性的な領域は，図 2 のコード中，同じ箇所でコメントアウトされた is_a_social_ordering(_,[W|_],L)に置き換えて実験したとき，社会的選択ルール生成に失敗する領域と完全に一致する．

最後に，3 種類の価値制限のうち，単峰性（sp）を満たす場合の多数決ルールを実験した．結果を図 6 と図 7 に分けて示す．この実験では，操作可能性（m），独裁性（d），市民主権の条件への違反（ncs）についても，この順に検証し，最初に検出されたものを領域の表示の後に記した．

図 6 と図 7 は，価値制限のうち，単峰性（sp）を満たす領域について，多数決 SCF ルールが，上記の 3 条件を同時に回避する（またそれゆえに Arrow の定理や Gibbard-Satterthwaite 定理の条件を満たす）ことが可能か否かについて調べ，可能と判明した領域は，すべてのプロフィールでの多数決の勝者（多数決 SCF）を表示した．各行は人 1 の各ランキングに対応する．

[2] 各 3 つ組みついて，3 条件のうちいずれかを満たすというのが価値制限の定義であり，図 5 のコマンドラインのように，sp, sc, s2 のいずれかの値を指定した value_restriction/1 の実行結果は，3つ組みは唯一である 3 代替案の場合でのみ，一致する．図 6 の実験についても同様である．4 代替案以上に拡張する場合，価値制限として実行するには，この変数を非束縛にする必要がある．

各行内では残りの 2 人のそのプロフィールの部分が,

#r2: 1111112222223333334444445555556666666

#r3: 1234561234561234561234561234561234561234

の順に並び，それぞれ対応する多数決投票の勝者を表していると理解されたい．図 6 と図 7 で，9 つの領域が該当し，それぞれ一意に多数決 SCF（および多数決 SWF）が定まる．またそれ以外の領域では，3 条件のいずれかに抵触する．ただし ncs と記されているものについては，操作不可能性と非独裁性まではクリアしている．

詳細は省くが，二分割可能性(s2)の場合，同様に実験すると，sp の場合と共通する 3 ランキングすべてと，1246，1356，2345 の，やはり 9 領域が該当する．それ以外の場合，および単谷性(sc)の場合，該当する領域は存在しない．

6. おわりに

本論文では，実際に PROLOG を用いて，3 人 3 代替案の場合の線形順序の個人選好に対して，価値制限を満たす許容的領域の多数決ルールを実験的し，また操作不可能性，非独裁性，市民主権性をチェックした．価値制限を満たす，つまり推移的であるだけでは，これらの条件をすべて満たすことできない場合があった．例えば単一ランキングの領域は，自明に独裁性，操作不可能性，かつ価値制限を満たす．また 2 ランキング以下の領域は市民主権性を満たしえない．もっとも，ほとんど意見の相違がない，いいかえればじゅうぶん選好が類似しているなら，たとえ形式上，独裁的あるいは市民主権に違反したとしても，そう問題はないかもしれない．

ところで数式・論理式で語られる社会的選択理論はかつて研究者たちの創造的思考だったものである．それだけを追っても新しい成果は生まれない．彼らは，定理発見のヒントとなる典型的ないくつかの例を作ったにちがいない．3 つの主張で述べたように，論理プログラミングではコード自体が，定式化の代替物＝モデリングとして読める上に，事例を自動生成できる．もちろん，理解不足やミスにより，コード自体に誤りが発生することがある．しかし大半は

コードの意味を論理式と比べ，コードを用いたさまざまな実験を通じて発見・修正可能である．紙に絵を書くことが有用な場合もあり，コードの集まりにおいて閉じていない，ロジックマイニングを通じて，ある程度，背後にあったはずの直観的な理解を得られるのではないか．

参考文献

Arrow, K.: *Social Choice and Individual Values*, Yale University Press, 1963. (The first edition was published in 1951.)

Arrow, K., Sen, A. and Suzumura, K.: *The Handbook of Social Choice and Welfare*, North-Holland, 2002.

Clocksin, W. F. and Mellish, C. S.: *Programming in Prolog: Using ISO Standard*, 5th edition, Springer Verlag, 2003.

Gaertner, W.: *Domain Conditions in Social Choice Theory*. Cambridge University Press, 2001.

Gibbard, A.: "Manipulation of voting schemes: A general result," *Econometrica*, Vol. **41**: 587-602, 1973.

May, K. O.: "A set of independent necessary and sufficient conditions for simple majority decision," *Econometrica*, Vol. **20**: 680-684, 1952.

Murakami, Y.: "Formal structure of majority decision," *Econometrica*, Vol. **34**(3): 680-684, 1966.

Inada, K.: "On the simple majority decision rule," *Econometrica*, Vol. **36**: 490-506, 1969.

Indo, K.: "An automated proof of the Arrow's theorem," mimeo, 2006. (http://www.us.kanto-gakuen.ac.jp/indo/wp/myswf.pdf)

犬童健良: "社会的選択問題のための論理プログラミング," mimeo, 2006. (http://www.us.kanto-gakuen.ac.jp/indo/wp/lpmsc.pdf)

奥野正寛・鈴村興太郎:『ミクロ経済学 II』, 岩波書店, 1988.

佐伯胖:『決め方の論理』, 東京大学出版会, 1980.

宇佐美誠:『決定』, 東京大学出版会, 2000.

Satterthwaite, M. A.: "Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions," *Journal of Economic Theory*, Vol. **10**: 187-217, 1975.

Sen, A.: "A possibility theorem of majority decisions," *Econometrica*, Vol. **34**(2): 490-506, 1966.

Sen, A.: *Choice, Welfare and Measurement*, The MIT Press, 1982.

Sen, A. and Pattanaik, P. K.: "Necessary and sufficient condition for rational choice under majority decision," *Journal of Economic Theory*, Vol. **1**: 178-202, 1969.

Ward, B.: "Majority voting and alternative forms of public enterprise," In J. Margolis (ed.), *The Public Economy of Urban Communities*. Johns Hopkins Press, Chapter 6, pp. 112-126, 1965.

```
set_of_alternatives([a,b,c]).
set_of_agents([1,2,3]).

alternative(X):-
  set_of_alternatives(A),
  member(X,A).

agent(J):-
  set_of_agents(N),
  member(J,N).

possible_ranking_0( r(1), [a,b,c]).
possible_ranking_0( r(2), [a,c,b]).
possible_ranking_0( r(3), [b,a,c]).
possible_ranking_0( r(4), [b,c,a]).
possible_ranking_0( r(5), [c,a,b]).
possible_ranking_0( r(6), [c,b,a]).

possible_ranking( R, O):-
   possible_ranking_0( R, O),
   is_admmisibile( R).

:- dynamic  admissible_rankings/1.

admissible_rankings(
[r(1),r(2),r(3),r(4),r(5),r(6)]
).
```

```
prefer_x_to_y(A,B, R):-
   possible_ranking( R, O),
   append( _,[A|C],O),
   member(B,C).

prefer_x_to_y_0(A,B, R):-
   possible_ranking_0( R, O),
   append( _,[A|C],O),
   member(B,C).

possible_ranking_of( J, R, O):-
   agent(J),
   possible_ranking(R, O).

profile_of_rankings( [R1,R2,R3]):-
   set_of_agents([_,_,_]), %3-person,
   possible_ranking_of( 1, R1, _),
   possible_ranking_of( 2, R2, _),
   possible_ranking_of( 3, R3, _).

profile_of_rankings_in_tuple( Rn):-
   profile_of_rankings( LR),
   tuple_to_list(Rn,LR).

tuple_to_list(A,[A]).
tuple_to_list((A,T),[A|L]):-
   tuple_to_list(T,L).
```

図 1. 3 代替案，3 人の投票者の場合の可能なランキング，許容的領域，プロフィール

```
% a generic constructor
auto_scf( FL,Property):-
   all_profile_of_rankings( L),
   auto_scf( FL, L, Property).
% initial stage of recursions
auto_scf( [],[], _).
% scf of simple majority
auto_scf( [R-> X | H], [R|Q], majority):-
   auto_scf( H,Q,majority),
   is_simple_majority_decision( R->W),
   member(X,W).
auto_scf( [R->W|H],[R|Q],majority_set):-
   auto_scf( H,Q,majority_set),
   is_simple_majority_decision( R->W).

is_simple_majority_decision( Rn->WS):-
   collect_social_preference(Rn,L),
   findall(W,
     is_Condorcet_winner(W,L),
%     is_a_social_ordering(_,[W|_],L)
   WS0),
   sort(WS0,WS).

collect_social_preference(Rn,L):-
   findall((X,Y),
    (
     distinct_ordered_pair((X,Y)),
     simple_majority( Rn,(X,Y))
    ),
   L).
```

```
simple_majority( (R1,R2,R3),(X,Y)):-
   findall(1, (
     member(R, [R1,R2,R3]),
     prefer_x_to_y( X, Y, R)
   ),Poll),
   length(Poll,Nxy),
   length([R1,R2,R3],N),
   N =< 2* Nxy.

is_Condorcet_winner(W,L):-
   alternative(W),
   \+ (
     distinct_ordered_pair((W,X)),
     \+ member((W,X),L)
   ).

is_a_social_ordering(R,O,L):-
   possible_ranking_0(R, O),
   \+ (
     prefer_x_to_y_0(X,Y,R),
     \+ member((X,Y),L)
   ),
   \+ (
     member((X,Y),L),
     \+  prefer_x_to_y_0(X,Y,R)
   ).

all_profile_of_rankings(L):-
   findall(Rn,
     profile_of_rankings_in_tuple( Rn),
   L).
```

図 2. 多数決ルールによる社会的選択(3 人の場合)

```
type_of_value_restriction( 'sp',
  agree_on( not_worst),
  'Single-Peakedness'
).
type_of_value_restriction( 'sc',
   agree_on( not_best),
  'Single-Cavedness'
).
type_of_value_restriction( 's2',
  agree_on( not_medium),
  'Separability into 2 Groups'
).
value_restriction( S):-
   forall(
     triple_of_alternatives((X,Y,Z)),
     value_restriction( S, _C, (X,Y,Z))
   ).
value_restriction( S, C, (X,Y,Z)):-
   type_of_value_restriction(S,A,_),
   A = agree_on(C),
   member(W,[X,Y,Z]),
   subtract([X,Y,Z],[W],[U,V]),
   forall(
     possible_ranking( R,_),
     value_restriction(S, C, (W,U,V), R)
   ).
value_restriction( sp, _, (X,Y,Z), R):-
   prefer_x_to_y( X, Y, R)
   ;
   prefer_x_to_y( X, Z, R).
value_restriction( sc, _, (X,Y,Z), R):-
   prefer_x_to_y( Y, X, R)
   ;
   prefer_x_to_y( Z, X, R).
value_restriction( s2, _, (X,Y,Z), R):-
   prefer_x_to_y( X, Y, R),
   prefer_x_to_y( X, Z, R).
value_restriction( s2, _, (X,Y,Z), R):-
   prefer_x_to_y( Y, X, R),
   prefer_x_to_y( Z, X, R).

triple_of_alternatives( (X,Y,Z)):-
   alternative(X),
   alternative(Y), X@<Y,
   alternative(Z), Y@<Z.
```

```
value_restriction_1( S):-
   forall(
     triple_of_alternatives((X,Y,Z)),
     (
      type_of_value_restriction(S, A),
      A=agree_on(C),
      alternative(W),
      member(W,[X,Y,Z]),
      forall(
        profile_of_rankings( RL),
        is_agreed_on( C, W,(X,Y,Z), RL)
      )
     )
   ).

is_agreed_on( C, W, (X,Y,Z), RL):-
   member((K,C),[
     (1, not_best),
     (2, not_medium),
     (3, not_worst)
   ]),
   member(W, [X,Y,Z]),
   \+ (
     member(R, RL),
     rank_in_list( K, W, [X,Y,Z], R)
   ).

rank_in_list( K, A, List, R):-
   possible_ranking( R,_),
   member( A, List),
   rank_in_list_1( K, A, List, R).

rank_in_list_1( 1, A, List, R):-
   ¥+ (
     member( B, List),
     prefer_x_to_y( B, A, R)
   ).
rank_in_list_1( 3, A, List, R):-
   ¥+ (
     member( B, List),
     prefer_x_to_y( A, B, R)
   ).
rank_in_list_1( 2, A, List, R):-
   ¥+ rank_in_list( 1, A, List, R),
   ¥+ rank_in_list( 3, A, List, R).
```

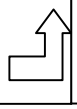
図 3. 価値制限（単峰性(sp)，単谷性(sc)，二分割可能性(s2)）の２通りのコード

```
value_restriction_by_latin_square:-
   \+ involves_latin_square(_,_,_).

involves_latin_square(type(1),(X,Y,Z),(R1,R2,R3)):-
   possible_ranking(R1,[X,Y,Z]),
   possible_ranking(R2,[Y,Z,X]),
   possible_ranking(R3,[Z,X,Y]).

involves_latin_square(type(2),(X,Y,Z),(R1,R2,R3)):-
   possible_ranking(R1,[X,Z,Y]),
   possible_ranking(R2,[Z,Y,X]),
   possible_ranking(R3,[Y,X,Z]).
```

```
% verification of the equivalence by simulations
?- auto_restricted_domain(H),
value_restriction_by_latin_square,
¥+ value_restriction(S).

No
?- auto_restricted_domain(H),
¥+ ¥+ value_restriction(S),
¥+ value_restriction_by_latin_square.

No
?-
```

図 4. ラテン方格を用いた場合の価値制限(上)とその同値性についての検証（下）

```
?- auto_restricted_domain(H),
\+ \+ value_restriction(S),
nl,forall(member(S,[sp,sc,s2]),
value_restriction(S)->write(S;' ')
;write('--; ')),write(H),fail.

--; sc; --; [r(1), r(2), r(3), r(4)]
sp; --; --; [r(1), r(2), r(3), r(5)]
sp; sc; --; [r(1), r(2), r(3)]
--; --; s2; [r(1), r(2), r(4), r(6)]
--; sc; s2; [r(1), r(2), r(4)]
--; sc; --; [r(1), r(2), r(5), r(6)]
sp; sc; --; [r(1), r(2), r(5)]
--; sc; s2; [r(1), r(2), r(6)]
sp; sc; s2; [r(1), r(2)]
sp; --; --; [r(1), r(3), r(4), r(6)]
sp; sc; --; [r(1), r(3), r(4)]
--; --; s2; [r(1), r(3), r(5), r(6)]
sp; --; s2; [r(1), r(3), r(5)]
sp; --; s2; [r(1), r(3), r(6)]
sp; sc; s2; [r(1), r(3)]
sp; --; s2; [r(1), r(4), r(6)]
sp; sc; s2; [r(1), r(4)]
--; sc; s2; [r(1), r(5), r(6)]
sp; sc; s2; [r(1), r(5)]
sp; sc; s2; [r(1), r(6)]
sp; sc; s2; [r(1)]
--; --; s2; [r(2), r(3), r(4), r(5)]
--; sc; s2; [r(2), r(3), r(4)]
sp; --; s2; [r(2), r(3), r(5)]
sp; sc; s2; [r(2), r(3)]
sp; --; --; [r(2), r(4), r(5), r(6)]
sp; --; s2; [r(2), r(4), r(5)]
sp; --; s2; [r(2), r(4), r(6)]
sp; sc; s2; [r(2), r(4)]
sp; sc; --; [r(2), r(5), r(6)]
sp; sc; s2; [r(2), r(5)]
sp; sc; s2; [r(2), r(6)]
sp; sc; s2; [r(2)]
--; sc; --; [r(3), r(4), r(5), r(6)]
```



```
--; sc; s2; [r(3), r(4), r(5)]
sp; sc; --; [r(3), r(4), r(6)]
sp; sc; s2; [r(3), r(4)]
--; sc; s2; [r(3), r(5), r(6)]
sp; sc; s2; [r(3), r(5)]
sp; sc; s2; [r(3), r(6)]
sp; sc; s2; [r(3)]
sp; sc; --; [r(4), r(5), r(6)]
sp; sc; s2; [r(4), r(5)]
sp; sc; s2; [r(4), r(6)]
sp; sc; s2; [r(4)]
sp; sc; s2; [r(5), r(6)]
sp; sc; s2; [r(5)]
sp; sc; s2; [r(6)]
sp; sc; s2; []

No
?-
```

図 5. 価値制限を満たす許容的領域を実験的に生成した

```
?- auto_restricted_domain(H),value_restriction(sp),nl,write(H),
auto_scf( F,majority),
(is_non_manipulable_scf(F)->true;(write(m),fail)),
(is_dictatorial_scf(F,_)->(write(d),fail);true),
(citizens_sovereignty(F)->true;(write(ncs),fail)),
show_scf_hr_0t(F),fail.

[r(1), r(2), r(3), r(5)]
scf:
r1=r(1)aaa-a-aaa-a-aab-a-------aaa-c-------
r1=r(2)aaa-a-aaa-a-aab-a-------aaa-c-------
r1=r(3)aab-a-aab-a-bbb-b-------aab-c-------
r1=r(4)-----------------------------------
r1=r(5)aaa-c-aaa-c-aab-c-------ccc-c-------
r1=r(6)-----------------------------------
[r(1), r(2), r(3)]ncs
[r(1), r(2), r(5)]ncs
[r(1), r(2)]d
[r(1), r(3), r(4), r(6)]
scf:
r1=r(1)a-aa-a------a-bb-ba-bb-b------a-bb-c
r1=r(2)-----------------------------------
r1=r(3)a-bb-b------b-bb-bb-bb-b------b-bb-c
r1=r(4)a-bb-b------b-bb-bb-bb-b------b-bb-c
r1=r(5)-----------------------------------
r1=r(6)a-bb-c------b-bb-cb-bb-c------c-cc-c
[r(1), r(3), r(4)]ncs
[r(1), r(3), r(5)]
scf:
r1=r(1)a-a-a-------a-b-a-------a-a-c-------
r1=r(2)-----------------------------------
r1=r(3)a-b-a-------b-b-b-------a-b-c-------
r1=r(4)-----------------------------------
r1=r(5)a-a-c-------a-b-c-------c-c-c-------
r1=r(6)-----------------------------------
[r(1), r(3), r(6)]
scf:
r1=r(1)a-a--a------a-b--b------------a-b--c
r1=r(2)-----------------------------------
r1=r(3)a-b--b------b-b--b------------b-b--c
r1=r(4)-----------------------------------
r1=r(5)-----------------------------------
r1=r(6)a-b--c------b-b--c------------c-c--c
[r(1), r(3)]ncs
[r(1), r(4), r(6)]
scf:
r1=r(1)a--a-a------------a--b-b------a--b-c
r1=r(2)-----------------------------------
r1=r(3)-----------------------------------
r1=r(4)a--b-b------------b--b-b------b--b-c
r1=r(5)-----------------------------------
r1=r(6)a--b-c------------b--b-c------c--c-c
[r(1), r(4)]ncs
[r(1), r(5)]ncs
[r(1), r(6)]ncs
[r(1)]d
```

図 6. 価値制限(sp)を満たす許容的領域において多数決ルールは操作不可能か？(前半)

```
[r(2), r(3), r(5)]
scf:
r1=r(1)-----------------------------------
r1=r(2)-------aa-a--ab-a--------aa-c-------
r1=r(3)-------ab-a--bb-b--------ab-c-------
r1=r(4)-----------------------------------
r1=r(5)-------aa-c--ab-c--------cc-c-------
r1=r(6)-----------------------------------
[r(2), r(3)]ncs
[r(2), r(4), r(5), r(6)]
scf:
r1=r(1)-----------------------------------
r1=r(2)-------a-aaa-------a-bcc-a-ccc-a-ccc
r1=r(3)-----------------------------------
r1=r(4)-------a-bcc-------b-bbb-c-bcc-c-bcc
r1=r(5)-------a-ccc-------c-bcc-c-ccc-c-ccc
r1=r(6)-------a-ccc-------c-bcc-c-ccc-c-ccc
[r(2), r(4), r(5)]
scf:
r1=r(1)-----------------------------------
r1=r(2)-------a-aa--------a-bc--a-cc-------
r1=r(3)-----------------------------------
r1=r(4)-------a-bc--------b-bb--c-bc-------
r1=r(5)-------a-cc--------c-bc--c-cc-------
r1=r(6)-----------------------------------
[r(2), r(4), r(6)]
scf:
r1=r(1)-----------------------------------
r1=r(2)-------a-a-a-------a-b-c-------a-c-c
r1=r(3)-----------------------------------
r1=r(4)-------a-b-c-------b-b-b-------c-b-c
r1=r(5)-----------------------------------
r1=r(6)-------a-c-c-------c-b-c-------c-c-c
[r(2), r(4)]ncs
[r(2), r(5), r(6)]ncs
[r(2), r(5)]ncs
[r(2), r(6)]ncs
[r(2)]d
[r(3), r(4), r(6)]ncs
[r(3), r(4)]d
[r(3), r(5)]ncs
[r(3), r(6)]ncs
[r(3)]d
[r(4), r(5), r(6)]ncs
[r(4), r(5)]ncs
[r(4), r(6)]ncs
[r(4)]d
[r(5), r(6)]d
[r(5)]d
[r(6)]d
[]d

No
?-
```

図 7. 価値制限(sp)を満たす許容的領域において多数決ルールは操作不可能か？（後半）