選好集計理論に基づく認知的モデリング：抜粋と補足[1]

犬童健良　関東学園大学経済学部経営学科

要約：選好集計理論([1][2][3])を認知的モデリングに応用し，前診断的プロセスにおける誤表象発生を論じた．例題として条件文推論の検証（選択課題）([4][5])を取り上げ，また論証ジレンマ[6]をその実用的スキーマとして再解釈し，前診断的プロセスの権利システム[7]としての側面に光を当てた．その上，PROLOG による実験により，満足化が反証主義[8]に取って代わりやすいことを厳密に示した．

選択課題[4][5]
認知科学の研究から，抽象的な条件文 p→q を検証する選択課題(selection task)の正答率はきわめて低く，多くの被験者は q を検査しようとする(確証バイアス)．一方以下のような許可スキーマ(permission schema)を与えると安定して正答率が高いとされる．

例）被験者に対して，１組のカードとリストが提示される．検証されるルール：

IF：カードに「搭乗」と記入されていたら，
THEN：リストの中に「コレラ」が記入されている．

その許可スキーマ:「あなたが空港の搭乗手続きを担当していると想定せよ．カードは搭乗予定者かどうかを示し，リストはその人の検疫記録である．乗客の安全を守るため，それをチェックしなければならない.」

本論文の解釈⇒許可スキーマはプライバシーや検査権限の設定．

[1] 本稿は経営情報学会 2006 秋季全国研究発表大会（兵庫県立大学 11/11）口頭発表における補足資料です（最終改訂 12/27）．予稿集論文中の証明の誤りを訂正し，PROLOG による実験結果を補足しました．また許可スキーマの安定性についての推測を追加しました．

経営情報学会 2006 年度秋季全国研究発表大会口頭発表用資料

モデル
条件文 p→q を検証するための実用的スキーマ（＝前診断的プロセス＋選択手続き）

表 1　実質含意に対する規範的な前診断的プロセス

observation data	satisficing set	complaint set	concerning set
P	{q}	{not q}	{not q}
Q	{p, not p}	ϕ	ϕ
Not p	{q, not q}	ϕ	ϕ
Not q	{not p}	{p}	{p}

観察可能なデータの集合 D＝{p, not p, q, not q}
満足化集合（satisficing set）S⊆D：D の部分集合でポジティブな証拠
愁訴集合（complaint set）C⊆D：同じくネガティブな証拠．
懸念集合(concerning set) K⊆S∪C：選択手続きを左右する部分．

権利システム[7]としての解釈
上例では K≠ ϕ ⇔検査を実行許可．より一般的には別紙 3 参照．

規範的スキーマと満足化

主張 1．（規範的スキーマ）反証主義者ないし希望の持てる懐疑主義[8]の懸念集合は愁訴集合に一致する．

主張 2．（満足化行動）エージェントが（反）満足化にしたがって検査を実行するのは，満足化集合と愁訴集合に対立する要素のペアがあって，かつ懸案集合がその一方の要素を含むときである．

論証ジレンマ[6] 表 2 のような命題の真偽についての多数決投票を考える．

表 2　論証ジレンマ．

member	p	p→q	q
Taro	Yes	Yes	Yes
Jiro	No	Yes	No
Hanako	Yes	No	No
majority	Yes	Yes	No

命題１. 表2はCondorcetの見出した多数決投票の循環と同等である．すなわち命題間の含意関係についての投票とみなせば，投票の背理である．（∵表 2 における各列の見出しは，それぞれ■→p，p→q，■→q と論理的に等しい．ただし■は恒真命題とする.）

命題２．表 2 は反証主義の手続きを構成する．（∵個人の投票を「真偽」ではなく，それを選択課題における「検査の必要性」とみなす．すなわち，各エージェントが満足化集合 S または愁訴集合 C の要素を主張し，社会はそれらを集計，懸念集合 K を決定する.）

命題３．含意関係が命題 1 の意味で選好関係とみなせるとき，単純多数決によって反証主義的な実用的スキーマが得られるのは，基本的に表 2 のパタンだけである．（∵付録４参照.）

命題４．反証主義のスキーマは確証バイアスを派生しやすい．またそれは安定している．（∵付録３参照.）

表 2'　論証ジレンマの変形.

Member	p	p→q	q
Taro	Yes	Yes	Yes
Jiro	No	Yes	No / 0*
Hanako	Yes	No	0*
Majority	Yes	Yes	0*

まとめ：多数決によって実現しうる反証主義は満足化に取って代わられる傾向がある．すなわち表 2'のような選好変化を契機に，満足化を実現する独裁的ないし寡頭的な権利配分に変わりやすく，またひとたび変化するともはや反証主義を導くことがない．一方，反証主義が出現し，かつ安定なスキーマは単独の拒否権者の場合だけである．これらのことから，許可スキーマが，(例えば封筒問題のような)その他の実用的推論スキーマとは異なり，安定して反証主義を実現しうる理由は，それが選択手続きのみならず，前診断的プロセスに影響を与えるゲームフォーム（効力関数）であり，より広範囲のプロフィールに対して有効だからであると推測される.

付録について：PROLOG を用いた実験

付録１．満足化により選択課題に対する前診断的プロセスが確証バイアスを発生させることを，PROLOG を用いて実験的に確かめる.

付録２．選好集計理論をモデリングし，シミュレーション実験を行うための PROLOG プログラムを紹介する.

付録３．上記プログラムを用い，命題４の Prolog による別証．すなわち表 2 およびその変形（表 2'）における権利配分を意味する単純ゲームを実験的に生成し，その安定性などを検証する.

付録４．同じく命題 3 の Prolog による別証．多数決が準推移性に違反する単純ゲームをもれなく生成，Salles[3]の循環的依存性条件への違反が表 2 と表 2`のパタンだけで必要十分であることを検証する.

参考文献

[1] Sen, A., *Choice, Welfare and Measurement*, MIT Press, 1982.

[2] Gaerter, W., *Domain Conditions in Social Choice Theory*, Cambridge University Press, 2001.

[3] Salles, M.: "Characterization of transitive individual preferences for quasi-transitive collective preference under simple games," *International Economc Review*, Vol. 17: 308-318, 1976.

[4] Evans, J.St.B.T., *Bias in Human Reasoning: Causes and Consequences*, Lawrence Erlbaum Associates Ltd., 1989.

[5] Holland, J.H., Holyoak, K.F., Nisbett, R.E. and Thagard, P.R., *Induction: Process of Inference, Learning, and Discovery*, MIT Press, 1986.（市川伸一ら訳,『インダクション』, 新曜社, 1991）

[6] Dietrich, F., "Judgment aggregation: (im)possibility theorems," *Journal of Economic Theory* Vol.126(1): 286-298, 2006.

[7] Peleg, B., "Effectivity functions, game forms, games, and rights," *Social Choice and Welfare* Vol.15: 67-80, 1998.

[8] Popper, K.R., Objective Knowledge, Clarendon Press, 1972.（森博訳,『客観的知識：進化論的アプローチ』, 木鐸社, 1974）

(付録１) ページ１／１

```
% the Wason's selection task
%-------------------------------------------------
card(no(1),'A', 3).
card(no(2),'D', 4).
card(no(3), 3, 'B').
card(no(4), 7, 'C').
（中略）
% possible observations for the selection task
data_item(p).
data_item(not(p)).
data_item(q).
data_item(not(q)).

% possible types of agent
type_of_agent(poppernian).
type_of_agent(ordinary).

% satisficing set: positive (or affirmative) evidences
satisficing_item(poppernian, p, q).
satisficing_item(poppernian, q, p).
satisficing_item(poppernian, q, not(p)).
satisficing_item(poppernian, not(p), q).
satisficing_item(poppernian, not(p), not(q)).
satisficing_item(poppernian, not(q), not(p)).
satisficing_item(ordinary, p, q).
satisficing_item(ordinary, q, p).
satisficing_item(ordinary, not(p), not(q)).
%satisficing_item(ordinary, not(q), not(p)).

% complaint set: negative (or denial) evidences
complaint_item(poppernian, p, not(q)).
complaint_item(poppernian, not(q), p).
complaint_item(ordinary, p, not(q)).
complaint_item(ordinary, q, not(p)).
%complaint_item(ordinary, not(p), q).
complaint_item(ordinary, not(q), p).

% the prediagnostic process
satisficing_set(T,A,B):-
   type_of_agent(T),
   data_item(A),
   findall( X, satisficing_item(T,A,X), B).
complaint_set(T,A,C):-
   type_of_agent(T),
   data_item(A),
   findall( X, complaint_item(T,A,X), C).
concerning_set(T,A,D):-
   type_of_agent(T),
   data_item(A),
   findall( X, concerning_item(T,A,X), D).
```



```
% concerning set: directly affects the choice procedure
concerning_item(poppernian,A,B):-
   complaint_item(poppernian, A, B).
concerning_item(ordinary,A,B):-
   satisficing_item(ordinary, A, B),
   complaint_item(ordinary, A, not(B)).
concerning_item(ordinary,A,B):-
   satisficing_item(ordinary, A, not(B)),
   complaint_item(ordinary, A, B).

% the choice procedure
choice_procedure(T,A,C):-
   data_item(A),
   possible_choice(C),
   rule_for_choice_procedure(T,A,C).
possible_choice(inspect).
possible_choice(not(inspect)).
rule_for_choice_procedure(T,A,inspect):-
   type_of_agent(T),
   concerning_item(T,A,_B),
   !.
rule_for_choice_procedure(T,A,not(inspect)):-
   type_of_agent(T),
   ¥+ concerning_item(T,A,_B).

% demo
/*
?- type_of_agent(T),choice_procedure(T,A,C),
nl,write(T:(A->C)),fail.

poppernian: (p->inspect)
poppernian: (not(p)->not(inspect))
poppernian: (q->not(inspect))
poppernian: (not(q)->inspect)
ordinary: (p->inspect)
ordinary: (not(p)->not(inspect))
ordinary: (q->inspect)
ordinary: (not(q)->not(inspect))

No
?-
*/
```

表 3　満足化を行う前診断的プロセスの一例

observation data	satisficing set	complaint set	concerning set
p	{q}	{not q}	{not q}
q	{p}	{not p}	{p}
not p	{not q}	ϕ	ϕ
not q	ϕ	{p}	ϕ

筆者の開発した Prolog プログラム sp06d.pl 中のコードの使用例をいくつか示す．その他，表 4 に述べられているようなモデリングが行える．

表 4　Prolog プログラム sp06d.pl で扱われる選好集計のモデリング

モデリング	人数	生成	主な項目
選好関係	2，3，…	可	線形順序，弱順序，準推移的順序等
領域制限	2，3，…	可	価値制限，循環的依存等
社会的選好	2，3	可	SWF，SDF（社会的選好とそれに基づく選択）
単純ゲーム	2，3	可	固有性，強性，弱性，コア，安定性等
効力関数	2，3	可	超加法性，極大性，凸性，コア，安定性等

●選好順序の生成 (r_0/5 -> r_x/5)

```
% demo: generating preference relations and orderings
/*
?- r_0(7,R,A,B,C).

R = [+, -, +]
A = [ (x, y): +, (x, z): -, (y, z): +]
B = [inconsistent, not(q-trans), complete]
C = [ (x, y), (z, x), (y, z)]

Yes
?- r_0(4,R,A,B,C).

R = [+, '0', +]
A = [ (x, y): +, (x, z):'0', (y, z): +]
B = [consistent, not(q-trans), complete]
C = [ (x, y), (x, z), (z, x), (y, z)]

Yes
?- linear_ordering.

---orderings:[1][3][9][19][25][27]
6 orderings has updated in r_x/5.

Yes
?- weak_ordering.

---orderings:[1][2][3][6][9][10][14][18][19][22][25][26][27]
13 orderings has updated in r_x/5.

Yes
?- quasi_ordering.

---orderings:[1][2][3][5][6][9][10][11][13][14][15][17][18][19][22][23][25][26][27]
19 orderings has updated in r_x/5.

Yes
?-
*/
```

●単純ゲーム

```
% code: simple game and the winning coalitions
:- dynamic win/2.

win( [], no).
win( [1], yes).
win( [2], no).
win( [3], no).
win( [1,2], yes).
win( [1,3], no).
win( [2,3], no).
win( [1,2,3], yes).

% demo
/*
?- verify_win.

game:[[1], [1, 2], [1, 2, 3]]
is proper
is weak with veto players:[1]
is essential

Yes
?-
*/
```

●エージェント社会の設定

```
%demo
/*
?- set_model(3-_,_), show_model.

agents:[1, 2, 3]
alternatives:[x, y, z]
coalitions:[[3], [2], [2, 3], [1], [1, 3], [1, 2], [1, 2, 3]]

Yes
?- set_model(2-_,_),show_model.

agents:[1, 2]
alternatives:[x, y, z]
coalitions:[[2], [1], [1, 2]]

Yes
?-
*/
```

単純ゲーム，効力関数，安定性：権利システムの数理的基礎として

- 単純ゲーム(simple game)は 1 人以上の可能なサブグループ（結託）のすべてのうち，単調性や固有性などの性質を満たすように特定の部分集合を勝利提携として選んだもの．権利（ないし権力）の配分を意味する．優越関係が成り立つ代替案同士の比較，すなわちその結託の内部のエージェントが強選好で合意すると，選好集計ないし社会的選択はそれに一致しなければならない．
- 特徴ゲームフォームは，任意の代替案の非空部分集合にかんして，単純ゲームと同様に，特定の結託の集まりを選び効力(effectiveness) を与える．効力関数(effectivity function)は，結託や代替案集合に関して単調な特徴ゲームフォームであり，また社会（グループ）全体はつねに任意の非空部分集合にかんして効力をもつ．単純ゲームはすべての非空の代替案部分集合において，同じ結託の集合が効力を割り当てられている特殊な効力関数である．
- 単純ゲームと効力関数は，優越関係を通じて共通のモデリングが可能であり，またそのコアは優越されない代替案の集合である．安定性は，任意の選好プロフィールに対してコアが非空であること．またそれは優越関係がサイクルをなさないことと同値である．

選択課題の実用的スキーマとしての論証ジレンマ：権利システムとしての安定性

- 本論文では前診断的プロセスの例題を，権利システム[8]として解釈することを試みた．懸念集合 K は情報断片 X⊆D に対して割当てられた検査の権利である．選択手続きは権利へのアクセス，行動集合 B={X を検査する，X を検査しない}の部分集合への写像である．反証主義の例では，観察データによらず，K=X∩C≠ φ⇔検査の権利（の実行）とされた．また P(Y)を集合 Y の非空部分集合族とすると，実用的スキーマは効力関数（特徴ゲームフォーム） f ：P(D)→→P(B)である．

PROLOG を用いた実験：命題４と命題３の証明

- 以下は PROLOG を用いて，まず命題４の別証，次いで命題３の別証を与える．
- ちなみに命題３の証明で用いた循環的依存性（価値制限と共に準推移的な社会的選好の必要十分条件をなす）は，厳密には固有ゲーム（の下での投票手続き）について述べられたものである．単純ゲームが固有(proper)であるというのは勝利提携の補集合は勝利提携になれないという条件をみたすときである．
- 命題４の別証．まず３人の社会での固有ゲームを生成し，社会的意思決定の準推移性への違反の有無を調べる．つぎにすべての単純ゲームを生成し，安定性を分析する．なおコマンドプロンプト？-からゴール入力される主な述語は以下のようである．

gen_win/1　現在のモデル（代替案とエージェント集合）の下で可能な単純ゲームを生成する
verify_win/1　現在のワークスペース上の単純ゲームモデルの諸性質を分析し表示する
is_unanimity_based_preference/4　結託内部の合意に基づく社会的選好を計算する
r_0/5, r_0(K,R,_,[_,q-trans,_],_)　生成した選好関係，その準推移性を満たすもの

- 準推移的でない社会的選好を生じない／生じる固有単純ゲームを生成する．

```
%Some proper games have no quasi-transitive orderings.
% N=[1,2,3] the society of taro-hanako-jiro case.
/*
?- gen_win(W),verify_win([yes,yes,_,V|_]),
¥+ (is_unanimity_based_preference(K,R,RN,Y),
(Y¥=true ;¥+ r_0(K,R,_,[_,q-trans,_],_))),
nl,write(W), tab(1), write(vetoers=V),fail.

[[1], [1, 2], [1, 3], [1, 2, 3]] vetoers=yes([1])
[[2], [1, 2], [2, 3], [1, 2, 3]] vetoers=yes([2])
[[3], [1, 3], [2, 3], [1, 2, 3]] vetoers=yes([3])
[[1, 2], [1, 2, 3]] vetoers=yes([1, 2])
[[1, 3], [1, 2, 3]] vetoers=yes([1, 3])
[[2, 3], [1, 2, 3]] vetoers=yes([2, 3])
[[1, 2, 3]] vetoers=yes([1, 2, 3])

No
?- gen_win(W),verify_win([yes,yes,_,V|_]),
¥+ ¥+ (is_unanimity_based_preference(K,R,RN,Y),
¥+ r_0(K,R,_,[_,q-trans,_],_)), nl,write(W), tab(1), write(vetoers=V),fail.

[[1, 2], [1, 3], [2, 3], [1, 2, 3]] vetoers=no
[[1, 2], [1, 3], [1, 2, 3]] vetoers=yes([1])
[[1, 2], [2, 3], [1, 2, 3]] vetoers=yes([2])
[[1, 3], [2, 3], [1, 2, 3]] vetoers=yes([3])

No
?-
*/
```

固有ゲームで社会的意思決定関数が準推移性に違反しないのは，独裁（上 3 パタン）か寡頭制および全会一致（下 4 パタン）．

多数決では強循環が発生しうる．また一つのペアの結託のみ無効化しても，まだ準推移性に矛盾しうる．

- 強ゲーム（つまり敗北提携の補集合は必ず勝利提携である）を生成し検査する．

```
/*
?- gen_win(W),verify_win([yes,yes,yes|I]),nl,verify_win,
verify_eff(S,EF),nl,write('core stability':S),nl,write(eff:EF),fail.

game:[[1, 2, 3], [1, 3], [1, 2], [1]]
is monotonic
is proper
is strong
is weak with veto players:[1]
is inessential with a dictator:1
core stability:[true, fail]
eff:[true, true, fail, true, true, true]
```

独裁は安定．すなわち，いかなるプロフィールにおいても独裁者(1)の提案を優越する別案を提案できる結託は存在しない．

```
game:[[1, 2, 3], [2, 3], [1, 2], [2]]
is monotonic
is proper
is strong
is weak with veto players:[2]
is inessential with a dictator:2
core stability:[true, fail]
eff:[true, true, fail, true, true, true]

game:[[1, 2, 3], [2, 3], [1, 3], [3]]
is monotonic
is proper
is strong
is weak with veto players:[3]
is inessential with a dictator:3
core stability:[true, fail]
eff:[true, true, fail, true, true, true]

game:[[1, 2, 3], [2, 3], [1, 3], [1, 2]]
is monotonic
is proper
is strong
is not weak
is essential
core stability:[fail, fail]
eff:[true, true, fail, true, true, fail]
```

単純多数決原理は不安定

```
No
?-
*/
```

● 強ゲーム以外の固有ゲームについての生成検査.

```
/*
?- gen_win(W),verify_win([yes,yes,no(T)|I]),nl,verify_win,
nl,write('is not strong':T),
verify_eff(S,EF),nl,write('core stability':S),nl,write(eff:EF),fail.

game:[[1, 2, 3], [1, 3], [1, 2]]
is monotonic
is proper
is weak with veto players:[1]
is essential
is not strong: ([2, 3], [1])
core stability:[true, fail]
eff:[true, true, fail, true, fail, fail]
```

花子(3)と次郎(2)の結託を無効化した場合は太郎(1)が唯一の拒否権者となり，安定である

```
game:[[1, 2, 3], [2, 3], [1, 2]]
is monotonic
is proper
is weak with veto players:[2]
is essential
is not strong: ([1, 3], [2])
core stability:[true, fail]
eff:[true, true, fail, true, fail, fail]

game:[[1, 2, 3], [1, 2]]
is monotonic
is proper
is weak with veto players:[1, 2]
is essential
is not strong: ([2, 3], [1])
core stability:[true, fail]
eff:[true, true, fail, true, fail, true]

game:[[1, 2, 3], [2, 3], [1, 3]]
is monotonic
is proper
is weak with veto players:[3]
is essential
is not strong: ([1, 2], [3])
core stability:[true, fail]
eff:[true, true, fail, true, fail, fail]

game:[[1, 2, 3], [1, 3]]
is monotonic
is proper
is weak with veto players:[1, 3]
is essential
is not strong: ([2, 3], [1])
core stability:[true, fail]
eff:[true, true, fail, true, fail, true]
```

花子(3)と太郎(1)の結託が唯一の拒否権者集合とした寡頭制の場合も，安定である

```
game:[[1, 2, 3], [2, 3]]
is monotonic
is proper
is weak with veto players:[2, 3]
is essential
is not strong: ([1, 3], [2])
core stability:[true, fail]
eff:[true, true, fail, true, fail, true]

game:[[1, 2, 3]]
is monotonic
is proper
is weak with veto players:[1, 2, 3]
is essential
is not strong: ([2, 3], [1])
core stability:[true, fail]
eff:[true, true, fail, true, fail, true]

No
?-
*/
```

- 命題３の別証．ステップ１～ステップ３によって示す．
- ステップ１：まず論証ジレンマの例における選好プロフィールのように，単純多数決原理の下で強循環する社会的選好（選好番号７）を生じるプロフィールをすべて求める．それらは表２のラテン方格において３人の名前を付け替えたものである．

```
/*
?- r_0(7,R,A,B,C).

R = [+, -, +]
A = [ (x, y): +, (x, z): -, (y, z): +]
B = [inconsistent, not(q-trans), complete]
C = [ (x, y), (z, x), (y, z)]

Yes
?- W = [[1, 2], [1, 3], [2, 3], [1, 2, 3]],
gen_win(W),is_unanimity_based_preference(K,R,RN,Y),
r_0(K,[+,-,+],_,_,B),
profile_in_numbers(RN,NR),nl,write(K:R;NR:RN),fail.

7:[+, -, +];[25, 9, 1]: ([+, -, -], [-, -, +], [+, +, +])
7:[+, -, +];[9, 25, 1]: ([-, -, +], [+, -, -], [+, +, +])
7:[+, -, +];[25, 1, 9]: ([+, -, -], [+, +, +], [-, -, +])
7:[+, -, +];[1, 25, 9]: ([+, +, +], [+, -, -], [-, -, +])
7:[+, -, +];[9, 1, 25]: ([-, -, +], [+, +, +], [+, -, -])
7:[+, -, +];[1, 9, 25]: ([+, +, +], [-, -, +], [+, -, -])

No
?-
*/
```

- ステップ２：次に同じく，単純多数決原理の下で，強循環はしないが，準推移的でない社会的選好（選好番号４）を生じるプロフィールをすべて求める．それらは３人の名前の付け替えによってそれぞれ同値である３つのパタンである．

```
/*
?- r_0(K,[+,'0',+],A,B,C).

K = 4
A = [ (x, y): +, (x, z):'0', (y, z): +]
B = [consistent, not(q-trans), complete]
C = [ (x, y), (x, z), (z, x), (y, z)]

Yes
?- W = [[1, 2], [1, 3], [2, 3], [1, 2, 3]],
gen_win(W),is_unanimity_based_preference(K,R,RN,Y),
r_0(K,[+,'0',+],_,_,B),
profile_in_numbers(RN,NR),nl,write(K:R;NR:RN),fail.

4:[+, 0, +];[22, 6, 1]: ([+, 0, -], [-, 0, +], [+, +, +])
4:[+, 0, +];[25, 6, 1]: ([+, -, -], [-, 0, +], [+, +, +])
4:[+, 0, +];[22, 9, 1]: ([+, 0, -], [-, -, +], [+, +, +])
4:[+, 0, +];[6, 22, 1]: ([-, 0, +], [+, 0, -], [+, +, +])
4:[+, 0, +];[9, 22, 1]: ([-, -, +], [+, 0, -], [+, +, +])
4:[+, 0, +];[6, 25, 1]: ([-, 0, +], [+, -, -], [+, +, +])
4:[+, 0, +];[22, 1, 6]: ([+, 0, -], [+, +, +], [-, 0, +])
4:[+, 0, +];[25, 1, 6]: ([+, -, -], [+, +, +], [-, 0, +])
4:[+, 0, +];[1, 22, 6]: ([+, +, +], [+, 0, -], [-, 0, +])
4:[+, 0, +];[1, 25, 6]: ([+, +, +], [+, -, -], [-, 0, +])
4:[+, 0, +];[22, 1, 9]: ([+, 0, -], [+, +, +], [-, -, +])
4:[+, 0, +];[1, 22, 9]: ([+, +, +], [+, 0, -], [-, -, +])
4:[+, 0, +];[6, 1, 22]: ([-, 0, +], [+, +, +], [+, 0, -])
4:[+, 0, +];[9, 1, 22]: ([-, -, +], [+, +, +], [+, 0, -])
4:[+, 0, +];[1, 6, 22]: ([+, +, +], [-, 0, +], [+, 0, -])
4:[+, 0, +];[1, 9, 22]: ([+, +, +], [-, -, +], [+, 0, -])
4:[+, 0, +];[6, 1, 25]: ([-, 0, +], [+, +, +], [+, -, -])
4:[+, 0, +];[1, 6, 25]: ([+, +, +], [-, 0, +], [+, -, -])

No
?- findall(NR,(is_unanimity_based_preference(K,R,RN,Y),
r_0(K,[+,'0',+],_,_,B),profile_in_numbers(RN,NR1),
sort(NR1,NR)),L1),sort(L1,L),nl,write(L).

[[1, 6, 22], [1, 6, 25], [1, 9, 22]]

Yes
?-
*/
```

- ステップ３：最後に，それらは循環的依存性に違反するパタン（表２の花子と次郎のいずれか一人か両名のｑについての選好を無差別に置き換えたもの）であることが確かめられる．

```
/*
?- preference_profile(RN),profile_in_numbers(RN,NR),
member(NR,[[1,9,25],[1,6,22],[1,6,25],[1,9,22]]),
nl,write(NR:RN),fail.

[1, 6, 22]: ([+, +, +], [-, 0, +], [+, 0, -])
[1, 9, 22]: ([+, +, +], [-, -, +], [+, 0, -])
[1, 6, 25]: ([+, +, +], [-, 0, +], [+, -, -])
[1, 9, 25]: ([+, +, +], [-, -, +], [+, -, -])

No
?-
*/
```