経済学紀要第30集(1) 投稿原稿

Prologによるナッシュ遂行のシミュレーション：実験システムの紹介

２００２年１２月１５日

犬童健良[1]

概要

筆者は、論理プログラミング言語 Prolog を用いて、ナッシュ遂行理論をモデリングする研究を行っている。本論文はその実験システムの概要を紹介した。モデリングの対象は、抽象的な社会選択環境を構成する選好領域、社会選択対応、および社会選択対応の諸性質、またナッシュ遂行理論においてこれに追加されるゲーム理論的メカニズム、ナッシュ均衡、およびナッシュ遂行の諸条件などである。モデリングとシミュレーションは、Prolog の論理プログラミングを用いてほぼ直接的に行えた。

内容

１．はじめに

[1] 関東学園大学経済学部経営学科助教授。Email:kindo@kanto-gakuen.ac.jp

筆者は、 論理プログラミング言語 Prolog を用いてナッシュ遂行理論を計算論的にモデリングし、シミュレーション実験する研究を行っている。本論文では筆者の開発した実験システムの概要を紹介する。

遂行理論 [4,5] は、社会選択理論やゲーム理論と共に、今日の経済学や経営科学において重要な基礎分野の一つである。また近年のエージェントモデル研究（図１参照）を基礎付ける意味もある。しかしこの分野は抽象度が高く、学習や教授において直観的な納得や厳密な理解への到達がむずかしい面があった。

論理プログラミング言語 Prolog はこの点で、具体的な操作（推論）と形式的な記述（知識表現）を橋渡しするのに都合が良い。実際、Prolog を使うことによって、選好順序や社会選択対応（SCC)、ナッシュ均衡、メカニズム（ゲームフォーム)などを、小規模ながらモデリングできる。

また、領域上の SCC を枚挙してナッシュ遂行の諸条件をチェックしたり、遂行可能な SCC を選んでシミュレーションによって確かめることも、原理的に可能である。ただし網羅的な生成やナッシュ遂行の逐次的テストが使えるのは、非常に小さな領域モデルに限られるが、実際に可能である。

論文の構成： Prolog による知的モデリングについては第３節で、また遂行理論などについて第３～５節でそれぞれ簡単に説明する。それぞれについて詳しい読者は、第６節まで飛ばしてよい。第６節でこれまでに開発された実験システムの概要を述べる。第７節と第８節で、本システムでできるモデリングとシミュレーションについて、例を交えて紹介する。第９節で開発経緯と謝辞などを述べる。第１０節で全体を統括する。

合理的選択モデル（最適化アプローチ）
実践指向（OR、経営工学、経営政策科学、制度設計工学など）
　→合理性の診断・処方、
理論指向（意思決定の数理と哲学・心理学研究）
　→最適化の数理、順序構造の数値表現、意識・態度の計測
知識処理モデル（人工知能アプローチ）
実践指向（知識工学、認知工学）
　→弱い AI：役に立つ知的機械の製造
理論指向（認知科学）
　→強い AI：知能・心のはたらきの解明
エージェントモデル（複雑適応系アプローチ）
実践指向（分散オブジェクト指向のシステム技術）
　→インターネットに適合するシステム開発
理論指向（新しい社会科学のアプローチ）

→合理的選択理論と人工知能を相補い融合する

図１．エージェントモデルを支える分野

２． Prologによる合理的エージェントモデル

論理プログラミング言語 Prolog の特色を以下にまとめる。また文法や理論背景等について付録２A と B にまとめた。

- ホーン節形式を格納する内部データベースを持つ。
- 導出原理と単一化アルゴリズムによる自動証明を実行する。
- 論理的知識表現とリスト（２進木）操作を融合 。
- 自動証明によりバックトラッキング（後戻り処理）を内蔵 。
- 宣言的意味と手続き的意味の両面性を獲得する。
- また合理的選択理論に沿ったエージェントモデルの構成は以下のようである。
- エージェントの集合
- 代替案（行動、選択肢、活動）の集合
- 選好順序（状態）または効用関数

以下にPrologによる知的エージェント、および合理的エージェントの簡単なモデリング例をいくつか示そう。（Prologの例１～５）

Prologの例１（知識ベース）

たとえばトリは例外的条件に該当しない限り飛べるという知識をルール節として、またスズメはトリであるという事実節としてそれぞれ登録する。スズメは飛べるかどうかをゴール節として問い合わせると、Prolog システムは Yes と答える。

```
トリ(スズメ). トリ(ペンギン). トリの例外(ペンギン).
飛ぶ(X) :- トリ (X), \+トリの例外(X).
？- 飛ぶ(スズメ). ［enter キー ］
Yes
```

記号\+は「失敗による否定」(nagation as failure)を表す。ペンギンはトリの例外として登録されているので、同様の問合せに対し、Prolog システムは No と答える。

```
？- 飛ぶ(ペンギン). ［enter キー ］
No
```

Prologの例２（集合論的操作）

知識ベースの例：

```
エージェント(名前(a),年齢(20),所得(0),身長(175)).
エージェント(名前(b),年齢(32),所得(800),身長(180)).
エージェント(名前(c),年齢(45),所得(1200),身長(168)).
お金がない (X,Z) :- エージェント(名前(X),_,所得(Z),_), Z < 10.
```

若者(X,Y) :- エージェント(名前(X),年齢(Y),_,_), Y < 35.
背が高い(X,H) :- エージェント(名前(X),_,_ ,身長(H)), H >= 175.
全称のテスト（forall/2） 「お金のない人は若者であるか？」
?- forall(お金がない(X,_Z) ,若者(X,_Y)). ［enter キー ］
Yes
ゴールの収集（ setof/3, bagof/3, findall/3 ） ^は記号存在
?- setof(X, A^若者(X,A), Y), setof(W, 背が高い(W,_M), Z). ［enter キー］
Y = [a, b]
Z = [a] （ß 条件に存在記号がないので_M も単一化される）Yes

Prologの例３（問題解決システム）

Prolog を使って人工的な知的エージェントの問題解決の機構（プランニング・システム）を次のように作ることができる。

移動(状態(s1),手段(a),状態(s2),コスト(2)).
移動(状態(s1),手段(b),状態(s2),コスト(5)).
移動(状態(s2),手段(c),状態(s3),コスト(1)).
現在(状態(s1)).
% 再帰的な目標達成プラン作成
目標（状態(G)) :-現在(状態(G)).
目標（状態(G)) :-
　目標（状態(S)), 移動（状態(S), _A,状態(G),_C).
?-目標（状態(s3)). ［enter キー ］
Yes

また目標到達までの手段系列（パス）を明示したり、そのコストを最小にするプログラムを改良することも容易であろう。

Prologの例４（合理的選択）

選好順序は推移的（ transitive)、反射的（ reflective）かつ完備（ complete）な代替案ペアの比較、すなわち弱順序である。無差別なペア a~b は、a>b かつ b>a のように順序を交換した連言で表し、強順序の場合は ¬（a~b）かつ a>b と定義する。以下の例では強順序を仮定する。

プログラム（事実節とルール節からなる）
エージェント(1). 代替案(a). 代替案(b). 代替案(c).
選好順序(エージェント(1), 状態(s1), [a, b, c]).
選好順序(エージェント(1), 状態(s2), [b, a, c]).
合理的選択(Agent,State,Choice):-
　　選好順序(Agent,State, Z, [Choice|_]).
ゴール節の実行例
?-合理的選択(1,s2,X).

```
X = b
Yes
```

Prologの例5 （ナッシュ均衡）

標準形ゲームとその最適反応の組（つまりナッシュ均衡）を求めるPrologプログラムの例およびその実行結果を示す。ただし、後でナッシュ遂行のシミュレーションで用いるものよりも、かなり単純である。

1＼2	b1	b2
a1	[1,1]	[2, 0]
a2	[0,2]	[-1, -1]

・プログラム

```
game( [player(1,act(a1)), player(2,act(b1)) ], payoffs([1,1])).
game( [player(1,act(a1)), player(2,act(b2)) ], payoffs([2,0])).
game( [player(1,act(a2)), player(2,act(b1)) ], payoffs([0,2])).
game( [player(1,act(a2)), player(2,act(b2))] , payoffs([-1,-1])).
nash( [S1,S2], [P1,P2] ):-
   game( [ player(1,act(S1)),  player(2,act(S2)) ], payoffs([P1,P2]) ),
     + ( game([ player(1,act(_X)),  player(2,act(S2)) ], payoffs([Px,_]), Px > P1 ),
     + ( game( [ player(1,act(S1)),  player(2,act(_Y))], payoffs([_,Py])), Py > P2 ).
```

・実行例

```
?- nash(A,B).
A = [a1,b1]
B = [1,1] ;
Yes
```

３．遂行理論

メカニズムデザイン論あるいは遂行理論は、理想的なエージェントの社会を以下のように抽象化し、何らかのゲームのルールの下で、自律分散したエージェントの活動の結果が、望ましい社会状態（社会選択対応；SCC）と一致するように、プランナーがそのようなゲームのルールを設計する問題を論じる。例えば、公共財の供給や、入札制度など、エージェントがウソをつく動機がある状況での経済学的応用がよく知られている。

- 社会選択環境の基本要素：
- 選択対象
- 社会のメンバーとその選好

（以上を「領域」(domain)モデルと呼ぶことにする。）

- 社会選択対応（SCC）　（社会選択規則（SCR）ともいう）
- メカニズム（計算システム、非協力ゲームなど）

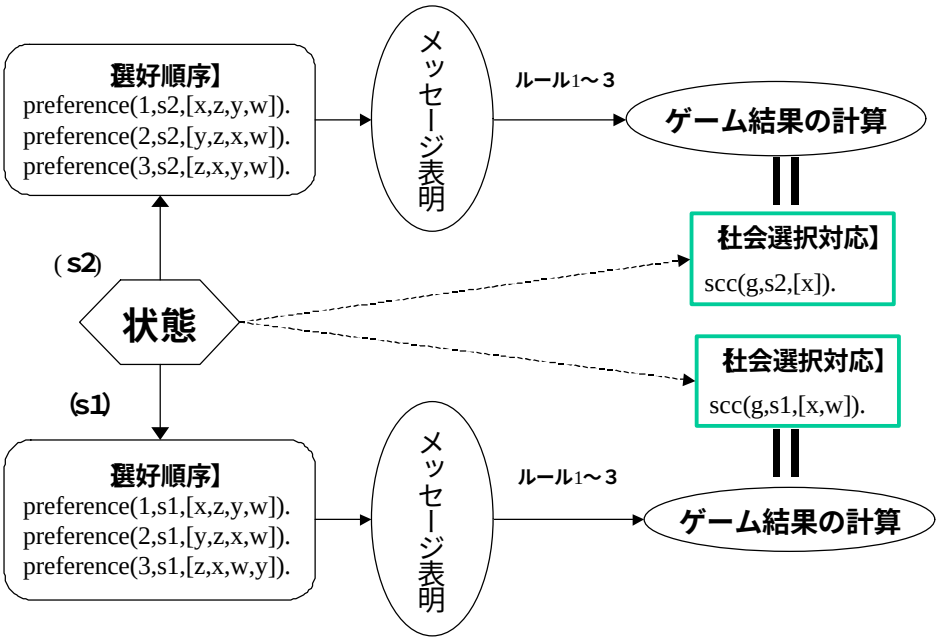


図２．ナッシュ遂行のイメージ

４．ナッシュ遂行

ナッシュ遂行理論で想定される状況は以下のようである。

個々人の選好は各エージェント同士はお互いに既知だが、プランナーからは観察できない。

プランナーは、各エージェントの選好の組に対して定義された社会選択対応（SCC;社会選択規則（SCR）ともいう）と、ナッシュ均衡が一致するようなメカニズム（ゲームのルール）を提案する。

メカニズムと選好組を合わせると、一つの非協力ゲームが定義される。ただしエージェントはナッシュ戦略、つまり最適反応を用いる仮定される。

そのナッシュ均衡集合につねに一致させることができる SCC は、遂行可能あるいはナッシュ遂行可能であるといわれる。

通常、メカニズムは２段階で実施される（図２）。まずメンバー各自が選好組や希望する代替案を吐くなど、プランナーに対してメッセージ送信を行う。プランナーはゲームフォームにしたがって選択結果を計算し、各エージェントに対して推奨する。

５．ナッシュ遂行定理

以下に（完全情報下の）ナッシュ遂行についての主な結果をまとめておく。

Maskin の得た先駆的な定理は次のようである。

３人以上の場合、ナッシュ遂行のための必要条件は、SCC の単調性である。（Maskin

単調性とも言う。その定義はモデル３を参照。）

３人以上では、誰も他の全員が最善と考える案を拒否できないこと、すなわち NVP（拒否権なし）の下で、単調性は必要十分条件である[4]。

３人以上では、Maskin（と Vind）のメカニズムを用いれば、単調かつ NVP である SCC をナッシュ遂行できる[4,1]。

しかし単調性だけは十分ではなく、また拒否権は必要条件ではない。より一般に、２人のケースを含めた必要十分条件は、Dutta と Sen[2]、および Moore と Repullo[6]によって 1990 年代初めに解かれた。 非制限領域（とその強選好部分）についての必要十分条件は、Danilov[1]によって示された。また彼らはそれぞれの条件を満たす任意の SCC を遂行するメカニズムを設計した。

その他、これまでに多くのナッシュ遂行の改良モデルが開発された。例えば、ナッシュ均衡では単独の逸脱のみ防げるが、結託した逸脱を許す強ナッシュ遂行についても解決済である。また仮想的遂行（および厳密遂行）と呼ばれる巧妙なメカニズムを用いれば、単調性にかかわらず任意の SCC を遂行できる。

非制限領域における必要十分条件

非制限領域はすべての可能な選好組からなる。ここでは非制限領域あるいはその強選好部分におけるナッシュ遂行の必要十分条件 [1] を述べる。ただし正式な定義ではなく、直観的な意味を説明する。より一般に制限領域での条件 [2,6] はその類似物であるが、やや複雑である。正式な定義などは文献に譲る。

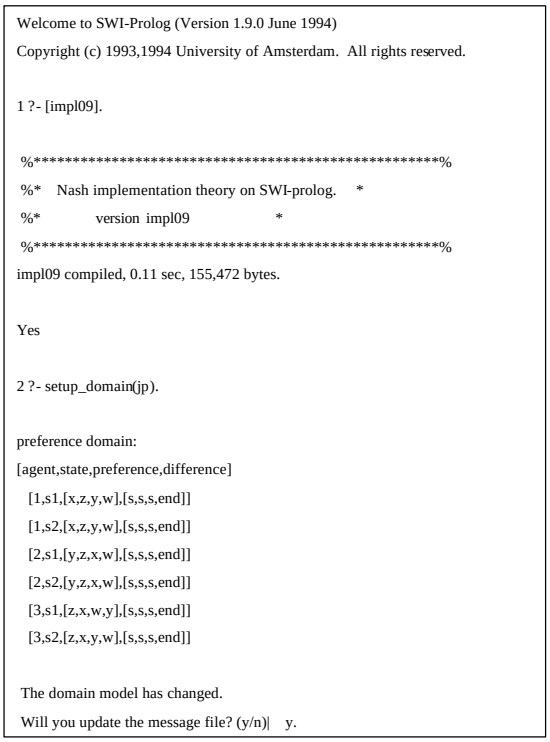
３人以上での必要十分条件は「本質的単調性」である[1]。制限領域でも十分条件であり、また非制限領域あるいは条件 D を満たす制限領域では必要条件となる[9]。２人の場合、これに「個人合理性」および「MR 特性」を加えた３条件である。後の２つは、「ブロック関係」を用いて定義される。これらについて、以下に非公式的な説明を試みる。

「本質的単調性」は Maskin による単調性を強めた条件である。つまり、ある SCC の結果が別の状態で SCC から外れてよいのは、選好逆転を起こす別の SCC 結果があって、かつそれは元の SCC 結果の劣位集合の本質的要素であった場合に限られる。

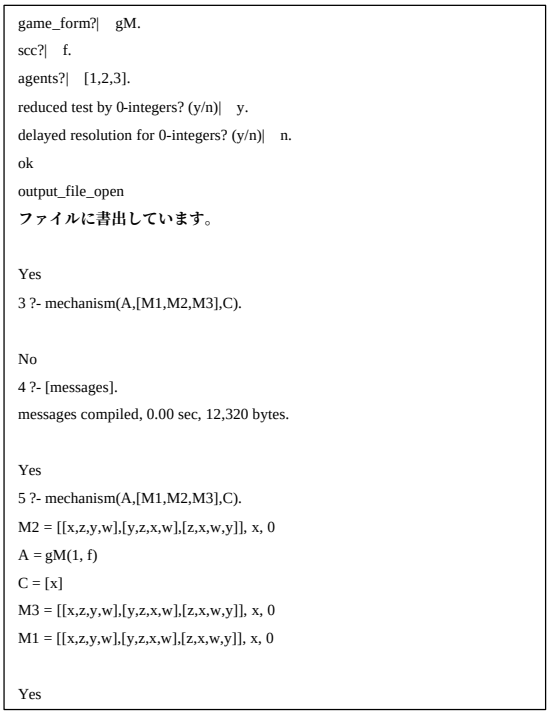
「ブロック関係」はおおむね次のように定義される。エージェントｉの選好Ｒｉがあって、そのとき他のエージェントたちの選好組に関わらず、代替案の集合 X が社会選択対応と交わらないとき、エージェントｉは集合 X をブロックするという。（NVP は全員がφのみブロックできることと等価である。）

「個人合理性」とは、誰かにブロックされるであろう劣った代替案を含まないこと。つまり個人合理的な SCC 結果は、各エージェント自身のその劣位集合をブロックしない。

「MR 特性」とは、両者がそれぞれブロックできないペアＸとＹの共通部分に共通最善案となる SCC 結果があること。



```
Welcome to SWI-Prolog (Version 1.9.0 June 1994)
Copyright (c) 1993,1994 University of Amsterdam.  All rights reserved.

1 ?- [impl09].

 %*****************************************************%
 %*   Nash implementation theory on SWI-prolog.    *
 %*            version  impl09                  *
 %*****************************************************%
impl09 compiled, 0.11 sec, 155,472 bytes.

Yes

2 ?- setup_domain(jp).

preference domain:
[agent,state,preference,difference]
   [1,s1,[x,z,y,w],[s,s,s,end]]
   [1,s2,[x,z,y,w],[s,s,s,end]]
   [2,s1,[y,z,x,w],[s,s,s,end]]
   [2,s2,[y,z,x,w],[s,s,s,end]]
   [3,s1,[z,x,w,y],[s,s,s,end]]
   [3,s2,[z,x,y,w],[s,s,s,end]]

 The domain model has changed.
 Will you update the message file? (y/n)|    y.
```



```
game_form?|    gM.
scc?|    f.
agents?|    [1,2,3].
reduced test by 0-integers? (y/n)|    y.
delayed resolution for 0-integers? (y/n)|    n.
ok
output_file_open
ファイルに書出しています。

Yes
3 ?- mechanism(A,[M1,M2,M3],C).

No
4 ?- [messages].
messages compiled, 0.00 sec, 12,320 bytes.

Yes
5 ?- mechanism(A,[M1,M2,M3],C).
M2 = [[x,z,y,w],[y,z,x,w],[z,x,w,y]], x, 0
A = gM(1, f)
C = [x]
M3 = [[x,z,y,w],[y,z,x,w],[z,x,w,y]], x, 0
M1 = [[x,z,y,w],[y,z,x,w],[z,x,w,y]], x, 0

Yes
```

図３．システム実行例１（立ち上げ、モデル設定、メッセージ一括処理）

６．実験システム概要

本節では Prolog の論理プログラミング技法を用いてナッシュ遂行理論をモデリング&シミュレーションするために、筆者が開発した実験システムの概要を述べる。

<u>モデリング</u>

対象は、社会選択環境を構成するA）選好領域、B）社会選択対応、およびC）社会選択対応の諸性質、またナッシュ遂行理論においてこれに追加されるD）ゲーム理論的メカニズム、E）ナッシュ均衡、および、F）ナッシュ遂行のテストである。

<u>シミュレーション</u>

主要な構成要素や条件については、それぞれのルールを用いて、ほぼ自己充足的に可能解の生成とシミュレーションを行える。また計算の効率化や作業の系統化を計るため、選好領域のモデル設定（setup_domain /1）や可能な SCC の枚挙（gen_test_sccs /3)、遂行シミュレーション(test_impl /0, /5)など、若干の専用ルーチンを追加した。（サンプル実行画面を図３～５に示す。）

<u>実験結果の保存と再利用</u>

シミュレーションで自動生成したメカニズム出力、SCC、最適反応については、それ

ぞれ節形式でファイルに保存し、Prolog システムに読み込んで再利用できる。表計算などで整理・加工してもよいが、多量データでの複雑な条件抽出などは Excel より早い。述語 results_to_file /0, /3 は、任意ゴールをバックトラックした結果を、節形式で指定ファイルに出力する。組み込み述語 tell /1 を用いて標準出力をリダイレクトする手もある。

```
7 ?- tests_for_scc(f,[1,2,3],O).

O = [no,no,no,no,bad(w),nvp,rvp,unan,no,no,no,neli,mlib]

Yes
8 ?- tests_for_scc(g,[1,2,3],O).

O = [mm,no,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]

Yes
9 ?- is_reversal(Agent,S1,S2,X,Y), X¥=Y.

Agent = 3
S1 = s1
S2 = s2
X = w
Y = y;

Agent = 3
S1 = s2
S2 = s1
X = y
Y = w;

No
```

```
6 ?- nash(full,x,s1,[M1,M2,M3],gM(1,f),h0,E,Is,NE).

 Now checking whether the message profile.

For state s1,outcome=x  [in,f],rule=1
and message profile is
    [[x,z,y,w],[y,z,x,w],[z,x,w,y]], x, 0
    [[x,z,y,w],[y,z,x,w],[z,x,w,y]], x, 0
    [[x,z,y,w],[y,z,x,w],[z,x,w,y]], x, 0
 no output file stream has opened.
wait..wait..wait..
 agent=1,P1s=[2],Czs=[x,z],Lcc=[w,x,y,z] <is_best_response>
 agent=2,P1s=[2],Czs=[x],Lcc=[w,x] <is_best_response>
 agent=3,P1s=[2],Czs=[x],Lcc=[w,x,y] <is_best_response>
Br_Members=[1,2,3]

This action profile is a Nash equilibrium.

NE = yes
E = environment([[1,2,3],[s1,s2],[w,x,y,z]],
 [3,2,4], [[[x,z,y,w],[y,z,x,w],[z,x,w,y]],[[x,z,y,w],[y,z,x,w],[z,x,w,y]]])
Is = [1,2,3]
M1 = [[x,z,y,w],[y,z,x,w],[z,x,w,y]], x, 0
M2 = [[x,z,y,w],[y,z,x,w],[z,x,w,y]], x, 0
M3 = [[x,z,y,w],[y,z,x,w],[z,x,w,y]], x, 0

Yes
```

図４．システム実行例２（SCCのテスト、ナッシュ均衡）

７．モデリング概要

本実験システムにおいて、Prolog によるモデリングを行なったナッシュ遂行理論の要素は、以下のようである。

- 選好領域（エージェント、状態、選好、選択対象）
- 社会選択対応（SCC）
- 劣位集合（LCC)、本質的集合（ESS）など
- 単調性、NVP、 パレート最適性など SCC の諸条件
- 最適反応とナッシュ均衡
- ナッシュ遂行のための各種メカニズム
- その他：部分集合、有限数列、モデル設定、ファイル出力等のユーティリティ

モデリング例（モデル１～１４)、およびユーティリティ１～２は付録３とした。

サンプル領域

本実験システムで、サンプルとして用意した領域とそのSCCを以下に挙げる。述語stock_of /1 に格納した選好作成用データを、setup_domain / 1 で展開するようにした。

領域名：jp，2 状態，4 代替案，3 人，参照：Jackson&Palfray(2001)，投票の例，SCC：[ｆ,f1,g,g1,and so on].

領域名：ks; ks1, 2 状態, 2 代替案, 2 人，参照：Rubinstein&Wolinsky(1994)，ソロモン王の例，SCC: [ks，ks1].

領域名：mr，4 状態，5 代替案，2 人，参照：Moore&Repullo [6] ，例２，SCC:mr.

領域名：md，3 状態，4 代替案，2 人，参照：Moore(1992)，脚注 91，SCC:mｄ.

領域名:y1; y2, 2状態, 3代替案, 2人，参照:Yamato [9] ，例２と３, SCC:[y1,y2].

領域名：ud22，4 状態，2 代替案，2 人，非制限領域の強選好部分，SCC:[udd1,udd2,urd1,…,urd4，umd1,…,umd5,par,par1,po，他自動生成].

領域名：ud32，8 状態，2 代替案，3 人，非制限領域強選好部分，SCC:自動生成.

領域名：ud23，36 状態，2 代替案，2 人，非制限領域強選好部分，SCC:自動生成.

メカニズム

本実験システムで利用できるメカニズムを以下に挙げる。これらはそれぞれ使用するメッセージ空間とそのメッセージ組の仕分け方と結果選択のためのルール群が異なる。またこれらのいくつかは部分的にルーレットを用いている。また述語rouletteを用意しているので、ルーレットだけのメカニズムを作るのは容易である。

メカニズム名：ｇDict，独裁的SCCを遂行するメカニズム，領域：一般，2人以上，ルール数:1.

メカニズム名：ｇM，Maskin-Vindの正準メカニズム，領域：一般，3 人以上，ルール数:3.

メカニズム名：ｇD，Danilovのメカニズム，領域：非制限，2 人，ルール数:3.

メカニズム名：ｇMR，領域：一般，Moore-Repulloのメカニズム，3 人以上，ルール数:3.

メカニズム名:ｇMR2，領域:一般，Moore-Repulloのメカニズム，2人，ルール数:6.

８．シミュレーション概要

モデリングされた述語は、一部を除き、ほぼ自己充足的に可能解を生成できる。また系統的に操作できるよう、若干のユーティリティとテストルーチンを追加した。以下に本実験システムでできる各種シミュレーションを分類してみよう。

- 選好領域（エージェント、状態、選好、選択対象）の設定
- 社会選択対応（SCC）の自動生成＆テスト
- 各種メカニズムに対するメッセージ集合の枚挙と保存
- ナッシュ均衡の生成
- ナッシュ遂行のチェック
- その他（部分集合や有限数列の枚挙、ルーレットなど）
- 遂行シミュレーション１～３(プログラム、シミュレーション例１～５）
- SCC 自動生成実験（プログラム、生成実験の例１～２）

模擬的制約処理と計算量の問題

ところで Prolog のルール節としてモデリングされたメカニズムを用いた、直接的なシミュレーションでは計算の効率が悪いため、本実験システムでは制約処理にならった工夫を施した。開発した述語の nash に対する test_nash と test_impl 内の check_on_scc, check_off_scc、および mechanism に対する ristrict_forms などがそれである。

例えば遂行シミュレーションでは、SCC 内外の状態と結果のペアについて、それぞれを出力するメカニズムへのメッセージを探してから、最適反応チェックをする。さもなくば nash 述語はメッセージを枚挙することになる。また SCC 内の結果については、正直報告を優先すれば、遂行可能な SCC についてはナッシュ均衡の発見に要する時間をさらに節約できるだろう。一つのメカニズム出力がナッシュ均衡であるかのテストは、数秒以内で済む。

もっとも、リソースは主に SCC 外のパタンでナッシュ均衡の不在を証明するために費やされる。また整数部分を０に基準化し、変異のときのみルーレットを使う不完全なシミュレーションでも、簡単な例題でも早くて１～２分、最も長くて２０分程度かかった。前述のように後にメカニズム出力については、前もっての一括検査方式に改めることにより、かなり高速化されたが、完全なシミュレーションでは依然計算量の壁が厚い。

９．開発の経緯、移植性、謝辞

ソースコードは全体で約 5 千行強、主な節形式が２０９種類（ただしアリティで区別せず)、計５８５個の節を、約１年かかって筆者単独で作成した。開発した述語の一覧と説明は付録にした。開発環境はノートＰＣとＰｒｏｌｏｇおよびテキストエディタ（秀丸）のみで、テストと一部の実験に研究室や教育用デスクトップＰＣを使用した。

開発と実験には SWI-Prolog version 1.9.0 [8] （Windows 版）を用いた。 同ソフトウェアはフリーの Prolog 処理系でありながら、軽快で、有用な述語が完備している。ただしメモリ管理等にややルーズな面がある。また同版は CMU の AI レポジトリからダウンロードできる。なお、この版の著作権は Jan Wielemaker とアムステル大学社会情

報学部が持つ。

本実験システムは、組み込み述語を除いて、C 言語などによる外部コードは作成していない。しかし他の処理系への移植性は未確認である。ただし商用処理系 IF-Prolog v.5.0B および GNU-Prolog については、nth1/3 や forall/2 や apply/2 や abolish/2 など、いくつかの不足する述語を別途作成すれば動作すると思われる。

・ 2002年10月20日に千葉で行われた第6回実験経済学コンファレンスおよび同11月17日に金沢で行われた経営情報学会秋季研究発表大会において、それぞれ本実験システムにかんする研究報告を行った。参加者の方々から有益なご指摘やご要望を頂いたことに感謝したい。

１０．おわりに

本論文で紹介した実験システムはその後も改定作業が続いているが、現時点で実験システムの概要をドキュメント化することにした。最新のバージョンは筆者のホームページを参照されたい(www.us.kanto-gakuen.ac.jp/indo/front.html)。

ナッシュ遂行を Prolog 化することの意義： ナッシュ遂行理論は、社会選択理論やゲーム理論をベースに、抽象化された合理的選好領域の上に展開される社会システムあるいは分権的メカニズムの数理である。その選好領域モデルとナッシュ遂行の集合論的条件は、Prolog の論理的知識表現とリスト処理に自然に適合する。このため理論の理解に役立つ。将来はメカニズム設計作業の自動化や、限界合理性の研究に役立つことを期待する。

本論文の実験システムでできたこと： 具体的にこの実験システムでできることは、合理的選択領域や社会選択対応のモデル化、ナッシュ遂行のための諸条件とメカニズムのモデル化とシミュレーション、および与えられた選好領域と任意の条件の下でナッシュ遂行可能な社会選択ルールを生成することなどである。

計算量の問題： 計算量ゆえの限界は、チェスプログラムが長年人間のチャンピオンに勝てなっかったのと似て、本実験システムの素朴なシミュレーションは、エキスパートの有する洞察レベルに至っていないからだともいえる。むしろ制約処理やデータマイニング技術の知られざる適用分野として、この分野の知識に注目できるだろう。

今後の課題： 一部のメカニズムはまだデバッグが十分でない。混合戦略についても省略されている。今後、経済学的環境や組織環境などの特殊領域のモデリングや、より発展したメカニズムの実装を含めて、改良の余地が多々残されている。また本システムを専門領域の知識ないし規範的モデルとする知的 CAI を構成し、非専門家の教育に使用できるようにしたい。

付録１．開発した主な述語の一覧

トップレベル	7
gen_test_sccs / 3	SCC自動生成＆一括テスト
setup_domain / 1	領域モデル設定とメッセージ検査
test_impl / 5	ナッシュ遂行シミュレーション
test_nash / 0, 7	ナッシュ均衡シミュレーション
tests_for_scc / 3	SCCの諸性質バッチテスト
update_message_file / 0	メカニズム出力悉皆検査
verify_messages / 6	現在のモデルで全メッセージをファイルに保存

選好領域	64
agent / 1	エージェント
agents / 1	エージェントの集合
all_agents / 1	リスト中のリストの全エージェント
all_preferences / 3	全選好組
alternative / 1	代替案
alternatives / 1	代替案集合
auto_preference / 3	自動生成する選好
auto_profile / 3	自動生成する選好組
auto_profiles / 3	自動生成する選好組集合
awk / 4	悲惨な結果集合
bad_outcome / 3	共通の悪い結果
bad_outcome1 / 3	共通の悪い結果
condition_D / 2	条件D（劣和）
decisive_pairs / 3	決定的要素のペア（peleg）
difference / 3	選好順序の差リスト
display_domain / 1	選好領域モデルの表示
domain_models / 1	利用可能な選好領域モデル
environment / 3	環境パラメーター一括
ess / 4	本質的要素の集合
gen_test_preference / 5	選好領域自動生成＆テスト
generate_preferences / 2	選好領域変異による生成
is_awkward / 5	悲惨な結果
is_essential / 4	本質的要素
is_MR_element / 5	MR要素（Danilov）
is_MR_elements / 4	MR要素の集合（Danilov）
is_prefer_to / 4	選好（二項関係）
is_reversal / 5	選好逆転関係
is_strict_prefer_to / 4	厳密選好（二項関係）
lcc / 3	劣位集合
maximal / 3	極大要素
maximal_set / 3	極大要素集合
mre / 5	Moore＆RepulloのMR要素
permutated_prefer_profile / 3	置換された選好組
ppp / 3	permutated_prefer_profile
prefer_profile / 2	選好組
prefer_profile / 3	選好組の集合
prefer_profiles / 3	選好順序
r_weak / 4	制限つき弱いエージェント
reversals / 6	選好逆転要素たち
scc_defined_state / 2	SCC定義済みの状態
scc_defined_states / 2	SCC定義済みの状態集合
set_B_star / 4	SjostromのB*
set_Bk / 4	SjostromのBk
set_C_star / 5	SjostromのC*
set_Ck / 5	SjostromのCk
set_dCk / 7	SjostromのC[k]-C[k-1]
slcc / 3	厳密な劣位集合
state / 1	状態
states / 1	状態集合
stock_of / 5	選好モデル（preference,difference）のストック
subset_of_agents / 2	エージェントの部分集合
subset_of_alt / 2	代替案の部分集合
succ / 3	厳密な上位集合
true_prefer_profile / 3	無矛盾な選好組
true_state / 1	ナッシュ均衡計算時の真の状態
two_person / 1	２人のエージェント
ucc / 3	上位集合
ud_preference / 5	非制限領域の選好
ud_profile / 3	非制限領域の選好組
ud32_preference / 1	３人２代替案の選好
ud32_preferences / 1	３人２代替案の選好組
unblocked / 4	ブロックされない集合
unblocked_pair / 3	ブロックされない集合ペア

社会選択対応（SCC）とその性質	40
clear_scc / 1	自動生成したSCC１のクリア
condition_M / 2	条件M（Sjostrom）
condition_M2 / 2	条件M２（Sjostrom）
condition_mju / 5	条件μ（Moore&Repullo）
condition_mju2 / 4	条件μ２（Moore&Repullo）
condorcet / 3	多数決対応
dictatorial / 2	独裁的SCC
display_scc / 3, 4	SCCの表示
ess_monotone / 1, 2	本質的単調性
generate_scc / 2	SCC自動生成（scc1または scc0）
has_condorcet_property / 2	多数決対応のテスト
has_MR_property / 2	MR特性のテスト
has_pareto_property / 2	パレート対応のテスト
i_rationals / 5	個人合理的集合
indifferent_to / 4	無差別関係
is_blocked_by / 4, 5	ブロック関係（Danilov）
is_decisive / 4	決定性（Peleg）
is_I_rational / 4, 5	個人合理性（Danilov）
is_neutral / 2	中立性
is_weak_parete_optimal / 3	弱パレート最適
is_wpo / 3	is_weak_parete_optimal
mlib / 2	Senの最小自由主義
monotone / 1, 2	Maskin単調性
neli / 2, 3	No EmptyLower Intersection
no_veto_power / 3	拒否権なし
no_veto_power0 / 3	拒否権なしの別モデル
no_veto_power1 / 3	拒否権なしの別モデル
nvp / 2	no_veto_power
nvp0 / 2	no_veto_power0
nvp1 / 2	no_veto_power1
pairwise_majority / 6	ペア多数決
partial_wpo / 3	部分的パレート最適
range / 2	SCCの値域
rvp / 2	制限付き拒否権
scc / 1	SCC（社会選択対応）
scc_tuple / 3	自動生成用SCC組
sccs / 1	SCCの集合
sub_condition_of_mju / 4	条件μの下位条件
subset_of / 3	任意の部分集合
unanimity / 2	全会一致
update_scc / 2	SCC自動生成での更新

メカニズム、ゲームと均衡解	0
best_response / 5	最適反応
br_member / 2	最適反応メンバー
br_result / 1	最適反応データ
check_br_of_profile / 4	最適反応のチェック
collect_br_members / 2	最適反応のエージェントをリストアップ
dictatorship / 4	（パートタイム）独裁権
game_form / 4	ゲームフォーム
game_forms / 2	ゲームフォーム集合
is_a_dictator / 3	独裁者
is_beta_blocked_by / 6	メカニズムによるブッロク
mechanism / 4	メカニズム
mechanisms / 1	メカニズムの集合
messages / 5	メッセージ組
mr_msg / 4	メカニズムｇM用メッセージ
mr_profile / 3	メカニズムｇM用選好組
mr2_msg / 4	メカニズムｇM２用メッセージ
mr2_profile / 3	メカニズムｇM２用選好組
mtest / 2, 4	メッセージ組
mtest_z0 / 4, 5	整数部分が０のメッセージ組
mutate / 6	メッセージの変異
nash / 9	ナッシュ均衡
restrict_forms / 2	ナッシュ均衡シミュレーションでの制約処理
roulette / 3	ルーレット モジュロn）
setup / 3	メカニズムのパラメータ設定
tt_gM_profile / 4	真実報告の ｇM選好組
update_state / 1	ナッシュ均衡のテストの中での状態更新
write_best_response / 2	エージェントの最適反応チェック表示
write_br_results / 2	最適反応チェック表示
write_message / 5, 6	メッセージ書き出し

メカニズム、ゲームと均衡解 30

best_response / 5	最適反応
br_member / 2	最適反応メンバー
br_result / 1	最適反応データ
check_br_of_profile / 4	最適反応のチェック
collect_br_members / 2	最適反応のエージェントをリストアップ
dictatorship / 4	《パートタイム》独裁権
game_form / 4	ゲームフォーム
game_forms / 2	ゲームフォーム集合
is_a_dictator / 3	独裁者
is_beta_blocked_by / 6	メカニズムによるブッロク
mechanism / 4	メカニズム
mechanisms / 1	メカニズムの集合
messages / 5	メッセージ組
mr_msg / 4	メカニズムｇM用メッセージ
mr_profile / 3	メカニズムｇM用選好組
mr2_msg / 4	メカニズムgM２用メッセージ
mr2_profile / 3	メカニズムｇM２用選好組
mtest / 2, 4	メッセージ組
mtest_z0 / 4, 5	整数部分が０のメッセージ組
mutate / 6	メッセージの変異
nash / 9	ナッシュ均衡
restrict_forms / 2	ナッシュ均衡シミュレーションでの制約処理
roulette / 3	ルーレット（モジュロn）
setup / 3	メカニズムのパラメータ設定
tt_gM_profile / 4	真実報告のｇM選好組
update_state / 1	ナッシュ均衡のテストの中での状態更新
write_best_response / 2	エージェントの最適反応チェック表示
write_br_results / 2	最適反応チェック表示
write_message / 5, 6	メッセージ書き出し
write_nash / 3, 4	ナッシュ均衡の書き出し
write_nash_equilibrium / 2, 3	ナッシュ均衡の書き出し
writeMessages / 2	メッセージ組の書き出し

遂行シミュレーション 14

check_off_scc / 6	遂行シミュ：SCC内のパタン
check_on_scc / 6	遂行シミュ：SCC外のパタン
clear_br_results / 0	遂行シミュ：BRデータを消去
close_output_file / 1	遂行シミュ：出力ファイルを閉じる
collect_statistics / 1	遂行シミュ：統計値の収集
disp_mr_ir / 2, 3	遂行シミュ：２人のときのSCC追加テスト
display_br_results / 3	最適反応のデータ表示
init_state / 0	遂行シミュ状態初期化
init_summary / 1	遂行シミュ要約初期化
open_br_file / 4	BRデータ書き出し用ファイルを開く
open_output_file / 2	外部出力ファイルを開く
stat_keys / 1, 2	実験統計値のキー
update_summary / 1	ナッシュ遂行シミュレーション中のデータ更新
write_statistics / 2	実験の統計値書き出し
write_summary / 1	実験結果要約の書き出し

集合操作 18

all_members / 2	全リスト中リストの要素
asc_nnseq / 2	昇順整数列
bag0 / 3	重複を許す有限集合
bag1 / 3	有限集合
counter / 3	リスト中の出現カウント
desc_nnseq / 2	減少整数列
list_projection / 3	リスト射影
multiple_subset_of / 3	重複部分集合
nth_of_permutation / 4	第ｎ座標の代替案置換
nth_poa / 3	nth_of_permutation
ordering / 3	順序
permutate_of_order / 2	置換された順序
permutation / 3	置換
permutation_of / 3	置換
poa / 2	permutate_of_order
powerset_of / 2	部分集合族
sea_multiple / 4	ゴール成功数の検査
sum / 2	足し算

外部ファイル入出力 3

results_to_file / 0, 3	ゴールバックトラック結果の保存
save_br_results / 2	最適反応チェック結果の保存
tell_test / 1	実行経過をファイルに出力

テストまたはシミュレーション制御 16

subset_query / 3	A⊆BならC=yesさもなくばC=no
tell_test_mju / 1	条件μのテスト経過をファイルに出力
tell_test_mju2 / 0	条件μ２のテスト経過をファイルに出力
tell_test_nvp / 1	NVPのテスト経過をファイルに出力
test_for_model / 3	遂行シミュ：SCCの性質テスト
test_gD / 3, 4	メカニズムｇDのテスト
test_gM / 5	メカニズムｇMのテスト
test_gMR / 3	メカニズムｇMRのテスト
test_gMR2 / 3	メカニズムｇMR２のテスト
test_irat / 3	個人合理性テスト
test_irat_n2 / 2, 3	２人の個人合理性テスト
test_mrp_for_unblocked_pair / 3	非ブロックペアにMR要素あり？
test_nvp_def / 0	NVPの異なる定義の同一性テスト
test_proposition / 3	Daniloの命題３テスト
test_sa / 2	test_subadditivity
test_subadditivity / 3	ブッロク関係の劣加法性テスト

開発記録 ･説明表示 10

edit_history / 0	本システム開発経緯表示
explain_me / 0	本システムの紹介
init_prompt / 0	プロンプト表示の変更
long_refer / 2	文献の長い表示
ref / 2, 7	文献データ
reference_list / 0	文献リストの表示
short_refer / 2	短い文献リスト表示
vshort_refer / 2	文献一覧（短い表示）
welcome / 0	ロード時ヘッダー表示
wn / 1, 2	write & nl

付録２A．Prologの構文、リスト記法など

Prologでは述語を述語名とカッコで表し、カッコ内には引数を任意個数カンマで区切って設定できる。

引数には個体記号、変数、リスト形式を使用できる。

個体記号：　小文字で始める文字列。あるいは数字。その他の文字列は引用符‘ ’で囲えばよい。また「親（親（X))」(＝X の祖父）のように変数を含んだ関数の使用も許される。

変数：　英大文字で始める。下線_で始めれば無名変数。

リスト形式：　任意個のアトムの列）も使える。たとえば[a,b,c,d]はリストだが、左端と残りのペアを縦棒で区切って[a|Y] 、Y=[b,c,d]と単一化できる。

事実は一つの述語で、ルールは頭部に述語、首記号:-で区切って右側に本体を書く。本体は

任意個の述語をカンマで区切って書ける。
本体の条件はカンマで区切られた連言を意味するが、セミコロン；で選言を表せる。例．英語科目:-英文法; 英会話.（⇒英語科目:-英文法.英語科目:- 英会話.のように２つの節に分けても良い。）
なおカット！はそれ以降のゴール失敗のとき、バックトラックを！以前まで遡及させない。
矢印 - >は If-Then 文として使える。

付録２B．Prologによる自動証明の基礎（導出と単一化）

・Prolog プログラムは、そのリテラルも節全体もすべてが真理値（true or false）を持つ。述語論理学の論理的意味と自動推論の原理（導出原理と単一化）に基づく手続き的意味は、SLD 導出にかんする完全性定理にかかわる。詳しくは成書[10]に譲る。また組み込み述語等は Prolog 処理系のマニュアル[8]を参考にされたい。
節形式はリテラルの選言（「または」を意味する論理結合子（∨）で結ばれたリテラルの列）である。ホーン節(Horn clause)は肯定リテラルが一つだけ許されている節形式であり、以下のような冠頭標準形に対応する。リテラルは引数を特定した述語と考えてよいが、否定がつく場合を含む。
ルール節 p(x):- q(x), r(x). の宣言的意味は「もし x が q と r という条件を同時に満たしているならば、p という条件も満たすはずである。」という主張に等しい。論理式では、q∧r → p あるいは p∨ not q∨ not r という形式で表せる（ただし全称∀が全体につくので唯一の変数 x は省略した)。 ∧（連言)、∨（選言)、not（否定)、→（条件）は標準的な論理学の約束にしたがう。
導出 (resolution)は論理学で言う「３段論法」と「背理法」の組合せである。単一化 (unification;ユニフィケーション)は、導出を行う際に変数に矛盾のない解を代入すること。背理法ではゴールが正しいことを証明するため、その否定を追加して矛盾を導くことを試みる。例えば「若者は味噌汁を好まない。」は、x は若者であるを p(x)、味噌汁を好むを q(x)とすると、p(y)→q(y) と書けるだろう。このときも p(a)が正しいなら３段論法により q(a)を得る。
変数の部分を無視すると、背理法にしたがって p→q という知識の下で q を証明しようとすれば、(p→q ）∧ not q ＝ （ not p∨ q) ∧ not q ＝ （ not p ∧ not q) ∨ (q∧ not q ）＝ （not p∧ not q ） ∨ （矛盾） ＝ not p ∧ not q となり、p という事実がもし別にあれば矛盾を導く。 きちんと書くと複雑なようだが、Prolog で導出をするときは単純な手続きの反復である。
ルール節は q(X):-p(X).となる。もし事実節 p(オタマジャクシ).や p(子供(カエル)).がデータベース内にあれば、ゴール q(X)は成功し、 X がオタマジャクシやカエルの子供とユニファイ（単一化）される。すなわちゴールと同じ頭部を持つ事実やルールを探して、事実なら成功し、ルールなら本体の条件を派生ゴールとして逐次実行する。
研究史：Prolog のしくみを司る導出原理 (Resolution principle) と単一化アルゴリズム (Unificaion algorithm)は J.R.Robinson の 1965 年の論文によってコンピュータによる自動定

理証明技術の基礎として確立されたものであり、後にColmerauer、Kowalskiによって Prologシステムの推論方式に採用された。1980年代には日本の第5世代コンピュータプロジェクトがPrologシステムをその基幹技術として注目し、自然言語処理、並列コンピュータ、LSI設計等への応用が意図された。

付録３．モデリング事例集

● モデル１．選好と社会環境

エージェント、代替案、状態、エージェントの選好（状態に依存すると仮定）および社会選択規則から成り立つ社会的選択の簡単な例題（３エージェント、４代替案、2状態）を Prologで記述する。なお強選好や無差別と混用するケースは別途工夫が要る。

```
/*   social environment including agents, preferences, states, and alternatives.  */
% 代替案、エージェント、状態の各集合
alternatives([w,x,y,z]). agents([1,2,3]). states([s1,s2]).
% エージェントの選好
preference(1,s1,[x,z,y,w]). preference(2,s1,[y,z,x,w]). preference(3,s1,[z,x,w,y]).
preference(1,s2,[x,z,y,w]). preference(2,s2,[y,z,x,w]). preference(3,s2,[z,x,y,w]).
prefer_profiles(Is,Ss,Rs):-  agents(Is), states(Ss),
   bagof(Rks,  S^( bagof(Rk,  I^(member(I,Is),preference(I,S,Rk)),Rks) ),  Rs).
% 環境パラメータの要約
environment([Is,Ss,As],[N,K,L],Rs):- !, length(Is, N), N = 3, agents(Is), states(Ss),
   length(Ss, K),  alternatives(As),   length(As, L),   prefer_profiles(Is,Ss,Rs).
```

● モデル２．社会選択対応

社会選択対応（Social Choice Correspondence）は社会選択関数（ふつうは１価）の多価の場合であり、エージェントの選好組（プロフィール）を代替案集合の非空部分集合に写像する。以下にいくつかの例を示す。

```
/* 社会選択対応 SCC; social choice correspondece, mapping F:S-->->X⊆A.  */
% a non-monotonic SCC: the example1 (Voting) of Jackson & Palfrey, p.8.
scc(f,s1,[x,z]).  scc(f,s2,[x]).  range(f,[x,z]).
% a monotone, but not ess_monotone SCC
scc(f1,s1,[x]).  scc(f1,s2,[x,y]).  range(f1,[x,y]).
% a monotone, but not ess_monotone SCC
scc(g,s1,[x,w]).  scc(g,s2,[x]). range(g,[x,w]).
% an ess_monotone SCC
scc(g1,s1,[x,w]).  scc(g1,s2,[x,y]).  range(g1,[x,w,y]).
```

```
% an ess_monotone SCC
scc(g0,s1,[x,y]).  scc(g0,s2,[x,y]). range(g0,[x,y]).
```

● モデル３．単調性と劣位集合

社会選択対応 SCC の単調性（Maskin 単調性とも呼ばれる）とは、次のことである。ある選好組（プロフィール)R、ないしそれを区別する状態において SCC に含まれる案（ｘ）は、別の選好組 R ’でも、SCC 内に選好逆転するエージェントと案がない限り外されない。劣位集合 (LCC)を使うと、単調性は次のように書ける。

（単調性）　x∈Scc（R）のとき Lcc（x,R;i）⊆Lcc （x,R ’;i）∀i →x∈Scc（R ’).

```
/*  Prolog による SCC の単調性（ Maskin  monotonicity ）のチェック.  */
monotone(F):-  agents(Is),  monotone(F,Is).
monotone(F,Is):-  scc(F,_,_), agents(Is),
   forall(
     ( state(S), scc(F,S,C), member(A,C), state(S1), scc(F,S1,C1), \+ member(A,C1)),
     ( member(I,Is),  lcc([I,S,_R],A,K1),  lcc([I,S1,_R1],A,K2), \+ subtract(K1,K2,[]))).
%  劣位集合 (lower contour set)
lcc([I,S,R],A,Lcc) :-  alternative(A),    agent(I),    state(S),    preference(I,S,R),
    findall(L, ( alternative(L), prefer_to(I,S,A,L)), Lcc).
```

● モデル４．選好逆転

劣位集合を使わず、ペアワイズの選好関係からより直接に選好逆転をチェックすると以下のようになる。これによって単調性が満たされないとき、原因となっている代替案とエージェントの選好を特定を追求できる。

```
/*  選好逆転をチェックする. */
prefer_to(I,S,X,Y):-  preference(I,S,Order),  rank(X,Order,J),  rank(Y,Order,K),  J =< K.
% ranking (for an weak ordered set)
rank(Z,List,Pos) :- nth1(Pos,List,Z).
%%%%%%% finding preference reversals.
is_reversal(I,S1,S2,X,Y) :-  prefer_to(I,S1,X,Y),  prefer_to(I,S2,Y,X).
reversals(I,S1,S2,X,A,Rvs):-  alternatives(A), subset_of_agents(Is,_N),  states(S),
   findall((I,X,S1,S2),  (
      member(I,Is),  member(X,A),  member(S1,S),  member(S2,S),
      is_reversal(I,S1,S2,X,_)
   ),Rvs).
```

● モデル５．本質的要素

代替案ｘ∈Y が全代替案の部分集合 Y（⊆A）においてエージェントｉにとって 「本質的である」というのは、ｘが社会選択対応に含まれ、かつ ｘの劣位集合 Lcc(i,x;R)が Y に含まれるｉの選好プロフィール R があるときである。すなわち社会選択対応の値域外であったり、あるいは選出されたときに劣位集合が焦点集合 Y からはみ出る場合はそのエージェントにとって本質的ではない。

```
% essential element
is_essential(A,X,[I,S,R],F):-
    (var(X) -> write('set X is unspecified.'),nl,fail);
    alternative(A),  state(S),  scc(F,S,C),  member(A,C),  agent(I),
    preference(I,S,R),  lcc([I,S,R],A,Lcc),
    forall(member(Y,X),alternative(Y)),   forall(member(Y,Lcc),member(Y,X)).
% essential set
ess(F,I,X,Ess):-  agent(I),  subset_of_alt(X,_),
   findall(A,  S^R^(alternative(A),is_essential(A,X,[I,S,R],F)), Y),  sort(Y,Ess).
```

● モデル６．本質的単調性

本質的単調性（Essential Monotonocity）は Danilov [1] が導入したもので、本質的集合（ESS)を使って、単調性を次のように強める。直観的に言えば、SCC から外される案を選好逆転するエージェントと別案があっても、SCC やその劣位集合の範囲外なら無効ということである。次のように定義される 。

（本質的単調性）

ｘ∈Scc（R）のとき Ess（Scc,;i, Lcc （x,R;i））⊆Lcc （x,R’;i）∀i→x∈Scc（R’）.

```
/* Prolog による SCC の本質的単調性（ Ess -monotonicity）のチェック.
ess_monotone(F,Is):-  scc(F,_,_), agents(Is),
% there is a reversal everywhere an ess_outcome retracted from scc.
   forall(
      (state(S), scc(F,S,C), member(A,C), state(S1), scc(F,S1,C1),  \+ member(A,C1)),
      (member(I,Is), lcc([I,S,_],A,K1),
       ess(F,I,K1,Ess),  lcc([I,S1,_],A,K2),  \+ subtract(Ess,K2,[])) ).
```

● モデル７Ａ．正準メカニズム（メイン）

Maskin-Vind のメカニズム [4, 3, 1] は、３人以上のケースでの十分条件を証明する。ただし SCC は単調とする。なお以降の各プログラムで環境変数を再記述しているのは、可能解を自己完結的に生成‐検証するためである。

```
/*  Maskin−Vind の正準メカニズムgM.  */
% the Cannonical Mechanism (Maskin-Vind mechanism):
mechanism(G,E,Msg,Z):-
    E = environment([Is,_Ss,_As],[_N,_K,_L],_Rs),   G = gM(_P,_Scc),
```

```
    setup(G,E),    mtest(gM, Msg,Is),    game_form(G,E,Msg,Z),!.
setup(G,E):-
    G = gM(P,Scc),    game_forms(gM,Ps),    member(P,Ps),    sccs(Fs),
    member(Scc,Fs),   subset_of_agents(Is,N),    states(Ss),    alternatives(As),
    E = environment([Is,Ss,As],[N,_K,_L],_Rs),    E.
% game forms.
game_forms(gM,[1,2,3]).
```

● モデル７Ｂ．正準メカニズム（メッセージ空間）

正準メカニズムのメッセージ空間は、各人の主張する選好プロフィール（全員の選好の組）とそのときの SCC 内の希望する代替案、それに加えて任意の自然数（N の剰余数）である。Maskin のメカニズムは当初この形ではなかったが、その後、改良が重ねられた［７］。

```
% message profile for N agents in mechanism gM.
mtest(gM, Msg, Is,Agents):-  agent(I), \+member(I,Is),    message(gM(_,_),Agents,I,M,true).
% message for N agents in mechanism gM.
message(gM(_,_),Is,I,(Rs,A,Z),Consistency):-
   environment([Is,_Ss,_As],[N,_K,_L],_Rks),  subset_of_agents(Is,N),  member(I,Is),
    (      Consistency = true
     -> (state(S),prefer_profile(Is,S,Rs))     ;  prefer_profile(Is,Rs)
    ),
   alternative(A),  asc_nnseq(Aseq,N),member(Z,Aseq).
 % ただし asc_nnseq/2 は１から N までの整数の昇順列
```

● モデル７Ｃ．正準メカニズム（ルール1）

正準メカニズムｇM には、３つのルールからなるゲームフォームが用いられる。最初のルールは整数部分を除き、全員の主張する選好プロフィールが一致している場合である。

```
%%%  the rule 1 of the Maskin mechanism gM. %%%%%%%%%%%%%%%%%%%%
%       整数ｍｉの部分を除き全員一致。（ｐｉ、ｃｉ）＝（ｐ、ｃ）かつｃ∈ｆ（≧、ｐ）==>  ｋ＝１。
game_form(gM(1,Scc),E,Msg,[C]):-
    E = environment([Is,_Ss,_As],[N,_K,_L],_Rs),    length(Msg,N),
    Msg = [(R,C,_Z1),(R,C,_Z2),(R,C,_Z3)],
    state(S),
    prefer_profile(Is,S,R),
    scc(Scc,S,Obj),
    alternative(C),
    member(C,Obj).
```

● モデル７Ｄ．正準メカニズム（ルール2）

第２のルールは一人を除いてルール１の条件が成立している場合を処理する。他の皆によって合意されたその人の選好では、その人自身にとって利益にならないとき、異議を唱えた人の独裁とする。それ以外は皆の合意にしたがってもらう。以下は３人のケースだけをモデル化しているが、ｎ人への一般化は容易である。また Lcc の部分を Ess に取り替えれば、非制限領域での完全なナッシュ遂行メカニズムになる [1]。

```
%%%% the rule 2 of the Maskin mechanism gM. %%%%%%%%%%%%%%%%%%
game_form(gM(2,Scc),E,Msg,[C]):-
      E = environment([Is,_Ss,_As],[N,_K,_L],_Rs),  length(Msg,N),
     (Msg = [(R1,C1,Z1),(R2,C2,Z2),(R2,C2,Z3)] -> (J is 1,true);
      Msg = [(R2,C2,Z1),(R1,C1,Z2),(R2,C2,Z3)] -> (J is 2,true);
      Msg = [(R2,C2,Z1),(R2,C2,Z2),(R1,C1,Z3)] -> (J is 3,true)),
      state(S),  prefer_profile(Is,S,R2), alternative(C1), alternative(C2), scc(Scc,S,Obj),
      member(C2,Obj), nth1(K,R2,R2J), preference(J,S,R2J), lcc([J,S,R2J],C2,Lcc),
      (member(C1,Lcc)-> C = C1; C = C2),  range(Scc,Range),member(C,Range).
```

● モデル７Ｅ．正準メカニズム（ルール3）

最後のルールはルール１およびルール２のいずれの条件も不成立のとき､ルーレットで独裁者を決める。Prolog ではバックトラックによって、前の２ルールが失敗すると、自動的にこのルールを試みることになる。メッセージの整数部分は人数の剰余であるから、合計の剰余＋１を計算してパートタイム独裁者を選ぶ。なお前述のように、この部分も１９７７年当初の Maskin の論文から変遷を経ている [7] 。

```
%%%% the rule 3 of the Maskin mechanism gM. %%%%%%%%%%%%%%%%%%
game_form(gM(3,_Scc),E,Msg,[C3a]):-
    E = environment([_Is,_Ss,_As],[N,_K,_L],_Rs),    length(Msg,N),
    Msg = [(_,C1,Z1),(_,C2,Z2),(_,C3,Z3)],!,
    SumZ is (Z1 + Z2 + Z3),
    K is SumZ mod N + 1,
    findall(C,member((R,C,Z),Msg), Cs),
    nth1(K,Cs,C3a).
```

● 正準メカニズムの解釈

正準メカニズムは正直報告の下でのナッシュ均衡を Scc と一致させるべく、階層的にデザインされた嘘つき均衡に対する「免疫システム」とみなせる。

全員の正直報告はルール１の下で均衡となる。　もし単独で虚偽報告しても、ルール２が作動し、有利な結果を誘導できない。

ルール２の下での嘘つき均衡の可能性は、ルール３の下でより有利な結果を誘導できるのでこれによって破壊される。

最後のルール３は内部でお互いに均衡の可能性を破壊しあう。

これらのことは、後で述べるシミュレーション結果（とくに Scc=g1 のとき）からも観察できる。

しかしこの免疫システムは単独での逸脱のみを防御している。（結託への免疫は強ナッシュ遂行によって解決されている。）また２人だけの場合、ルール２やルール３が機能しない。

● モデル８．ナッシュ均衡

ナッシュ均衡とは各エージェントがそれぞれナッシュ戦略を用いているときのゲームの結果、つまり後で説明する最適反応の組である。

```
/*  ナッシュ均衡. mechanism gM (N>=3). */
nash(C,S,Msg,G,H,E,Result):-   E = environment([Is,_,_],[_N,_,_],_),  E,  G =.. [GF,P,Scc],
    reversions(V), member(H,V),  sccs(Fs),    member(Scc,Fs), !,    update_state(S),
    alternative(C), mtest(GF, Msg,Is),  mechanism(G,E,Msg,[C]), write_message(S,C,P,Msg),
    findall([I,P1s,Czs,Lcc,Br],
     ( member(I,Is),  best_response(GF,I,Is,S,C,[Msg,Scc,E],[P1s,Czs,Lcc],Br),
       BrRecord = [Br,S,C,I,GF,P,Scc,Is,Msg,P1s,Czs,Lcc],  assert(br_result(BrRecord)),
       write_nash(I,[P1s,Czs,Lcc],Br) ),Br_Results),
   findall(I, (X = [I,P1s,Czs,Lcc,Br], member(X, Br_Results),Br = yes),  Br_Members0),
     sort(Br_Members0,Br_Members),  write_nash_equilibrium(Br_Members,Is),
    ( Br_Members = Is   ->   Result=yes  ; Result=no).
update_state(S):- state(S), forall(true_state(S1),retract(true_state(S1))), assert(true_state(S)).
```

● モデル９．最適反応

各人の行動が最適反応になっているかチェックするには、戦略変更(mutate/5)によって得られるメカニズムの可能な出力の集合 Czs （可達集合；attainable set とも呼ばれる）が、元のメカニズム出力の劣位集合 Lcc に包摂されていることを確かめればよい。

```
%%% checking best response via Lcc.
best_response(GF,I,Is,S,C,[Msg,Scc,E],[P1s,Czs,Lcc],Br):-
    E = environment([Is,_Ss,_As],[_N,_K,_L],_Rs),   E,   state(S),    sccs(Fs),
    member(Scc,Fs),  alternative(C),   game_forms(GF,Forms),   mtest(GF, Msg,Is),
    findall((P1,Cz),
       ( mutate(GF,I,Is,Msg,Mz),  member(P1,Forms),  alternative(Cz),
         G =.. [GF,P1,Scc],   mechanism(G,E,Mz,[Cz])
       ),     PCzs1),
    sort(PCzs1,PCzs),
    findall(P1,Cz^(member((P1,Cz),PCzs)),P1s1),  sort(P1s1,P1s),
    findall(Cz,P1^(member((P1,Cz),PCzs)),Czs1),  sort(Czs1,Czs),
```

```
    lcc([I,S,_],C,Lcc),   (subset(Czs,Lcc)-> Br = yes; Br = no).
```

● モデル１０．戦略の変異

元と同じメッセージの場合は除く。

```
% mutated profile where unilateral deviation for a single agent occured
mutate(GF,J,Is,Msg,Mz):-
   subset_of_agents(Is,N),   length(Is,N),
   mtest(GF, Msg,Is),length(Msg,N),
   mtest(GF, Mz,Is),length(Mz,N),   nth1(Jth,Is,J),
%
   subtract(Is,[J],Isz),
   forall(member(K,Isz),
      (  nth1(Kth,Is,K),   nth1(Kth,Msg,Mk),   nth1(Kth,Mz,Mk)  )
   ),
   nth1(Jth,Msg,MJ),
   nth1(Jth,Mz,MJz),
   MJz \= MJ.
```

● ユーティリティ１

```
/* some basic operations */
% projection(3rd ar) of vector(2nd ar) using a sequence of digits(1st ar).
list_projection([],[],[]).
list_projection([X|Y],[_A|B],C):-   X = 0,  list_projection(Y,B,C).
list_projection([X|Y],[A|B],[A|C]):-  X = 1,  list_projection(Y,B,C).
% count frequency of occurence of the specified value of variable, M.
counter(N,M,L):-   length(L,_),   findall(M,member(M,L),Mx),  length(Mx,N).
seteq(X,Y,L):-   length(X,L), length(Y,L), forall(member(Z,X),member(Z,Y)),
    forall(member(Z,Y),member(Z,X)).
subset_of(A,N,As):-
    length(As,L),  length(D,L), list_projection(D,As,B),  length(B,N),   sort(B,A).
%orderings with N elements
ordering([],_A,0).
ordering([C|B],A,N):-length([C|B],N), ordering(B,A,N1), member(C,A),\+member(C,B).
```

● ユーティリティ２

```
%subset_of(A,N,As):-multiple_subset_of(A,N,As).
multiple_subset_of([],0,_):-!.
multiple_subset_of([X|A],N,As):-
```

```
   length([X|A],N), multiple_subset_of(A,N1,As), member(X,As), N is N1 + 1.
powerset_of(X,A):-
   length(A,_), findall(Y,N^subset_of(Y,N,A),X1), sort(X1,X),!.
% descending natural number sequence less than N.
desc_nnseq([],N):-N<0,!.
desc_nnseq([0],1).
desc_nnseq([A|Q],N):-  A is N - 1,  length(Q,A),  desc_nnseq(Q,A).
asc_nnseq(Aseq,N):-desc_nnseq(Dseq,N),sort(Dseq,Aseq).
% sum
sum([],0).
sum([X|Members],Sum):-  sum(Members,Sum1),  number(X),  Sum is Sum1 + X.
```

● モデル１１．ブロック関係

```
/* blocking relation, individual rationality, and MR-property (Danilov,1992).  */
% agent I blocks set B if there is R(I) s.t. intersection(Scc([R(I),R(-I)]), B) is empty.
is_blocked_by(X,J,S,RJ,Is,F):-
    alternative(X),   subset_of_agents(Is,_N),   nth1(K,Is,J),   state(S),
    preference(J,S,RJ),   scc(F,S,_SccVal0),
    forall(  (state(S1),   prefer_profile(Is,S1,R1),  nth1(K,R1,RJ)),
      (scc(F,S1,SccVal),  \+member(X,SccVal))).
% maximal blocked set by each agent
blockings(Bs,J,Is,F):-
    sccs(Fs),  member(F,Fs),  subset_of_agents(Is,_),  member(J,Is),
    findall(X,  (preference(J,S,RJ), is_blocked_by(X,J,S,RJ,Is,F)),  Bs0),
    sort(Bs0,Bs).
```

● モデル１２．個人合理性

```
% individual rationality (individually rational outcomes)
i_rationals(As,S,Rn,Is,F):-
    scc(F,S,_Scc),   subset_of_agents(Is,_),   prefer_profile(Is,S,Rn),
    findall(A,  ( alternative(A),
        forall(member(J,Is),
          ( nth1(K,Is,J),  nth1(K,Rn,RJ), is_I_rational(A,[J,S,RJ],Is,F)))
         ),  As1),  sort(As1,As).
is_I_rational(A,[J,S,RJ],Is,F):-
    alternative(A),   preference(J,S,RJ),   subset_of_agents(Is,_N),
    lcc([J,S,RJ],A,Lcc),  scc(F,S,Scc),  \+blockings(Lcc,J,_S1,_RJ1,Is,F).
```

```
test_irat_n2(F,[I,J]):-
    sccs(Fs),  member(F,Fs), two_person([I,J]),
    forall(state(S), (scc(F,S,V),i_rationals(A,S,_R,[I,J],F),subset(V,A) )).
```

● モデル１３．MR特性

```
/*  Moore-Repullo property for two person case */
% MR-element
is_MR_element(A,[X1,X2],[[J1,J2],[R1,R2],S],F):-
    alternative(A),   scc(F,S,C), member(A,C),   prefer_profile([J1,J2],S,[R1,R2]),
    lcc([J1,S,R1],A,X1),   lcc([J2,S,R2],A,X2).
% check MR-property for scc
has_MR_property(F,[J1,J2]):-
    sccs(Fs), member(F,Fs),  two_person([J1,J2]),
    blockings(B1,J1,[J1,J2],F), blockings(B2,J2,[J1,J2],F),
    wn([for,[J1,J2],blockings,B1,B2, under_scc,F]),
    findall([X1,X2],   (  subset_of_alt(X1,_N1),  subset_of_alt(X2,_N2),
      \+subset(X1,B1),   \+subset(X2,B2)  ), X12),
    forall( member([X1,X2],X12),
     (wn([X1,X2]),   is_MR_elements(MR,[X1,X2],[J1,J2],F),    MR \= [] )).
```

● モデル１４Ａ．Danilovのメカニズム（メイン）

2エージェントのときScc F のナッシュ遂行の必要十分条件に対応するDanilov [1] のメカニズムをモデル化する 。

メッセージ空間は代替案の部分集合

Case 1: 1がX2をブロックせず、2がX1をブロックしないとき、ルーレット(modullo game)によってMR集合MR(F;X2,X1)≠[]から結果を選ぶ。

Case 2: iがXjをブロックし、jがXiをブロックしないとき、その本質的集合Ess(F;i,Xi)≠[]に含まれる結果についてjの独裁を認める。

Case 3: 1がX2をブロックし、2がX1をブロックするとき、ルーレット(modullo game)によってrange(F)から結果を選ぶ。

以下にDanilovメカニズムｇDのメッセージ空間とゲームフォームを記述する。

```
% message space for Danilov's mechanism
mtest(gD, Msg,Is):- two_person(Is),  Msg = [(X1,Z1),(X2,Z2)],
    subset_of_alt(X1,_),    member(Z1,[0,1]),
    subset_of_alt(X2,_),    member(Z2,[0,1]).
```

● モデル１４Ｂ．Danilovのメカニズム（ケース1）

```
two_person([J1,J2]):-  agents(Is),   member(J1,Is),member(J2,Is),J1 < J2.
%%%%%% the forms for the mechanism
game_form(gD(1,Scc),E,Msg,[C]):-
    E = environment([[J1,J2],_Ss,_As],[2,_K,_L],_Rs),
    two_person([J1,J2]),
    Msg=[(X1,_),(X2,_)],!,
    \+is_blocked_by (X2,J1,[J1,J2],Scc),
    \+is_blocked_by (X1,J2,[J1,J2],Scc),
     is_MR_elements(Y,[X2,X1],[J1,J2],Scc),wn([mr,Y]),
     SumZ is Z1 +  Z2,
     K is SumZ mod 2 + 1,
     dictatorship(K,_S,Y,C).
```

● モデル１４C． Danilovのメカニズム（ケース2)

```
game_form(gD(2,Scc),E,Msg,[C]):-
    E = environment([[J1,J2],_Ss,_As],[2,_K,_L],_Rs),
    two_person([J1,J2]),
    Msg=[(X1,_),(X2,_)],!,
  %  mtest(gD, Msg,[J1,J2]),!,
   (  (is_blocked_by(X2,J1,[J1,J2],Scc),
        \+is_blocked_by (X1,J2,[J1,J2],Scc), J0=J2 , X0=X1);
      ( \+is_blocked_by (X2,J1,[J1,J2],Scc),
         is_blocked_by (X1,J2,[J1,J2],Scc), J0=J1, X0=X2) ),
    ess(Scc,J0,X0,Ess0),
    member(C,Ess0),
    dictatorship(J0,_,Ess0,C).
```

● モデル１４D． Danilovのメカニズム（ケース3)

```
game_form(gD(3,Scc),E,Msg,[C]):-
    E = environment([[J1,J2],_Ss,_As],[2,_K,_L],_Rs),
    two_person([J1,J2]),
    Msg=[(X1,_),(X2,_)],!,
  %  mtest(gD, Msg,[J1,J2]),!,
    is_blocked_by (X2,J1,[J1,J2],Scc),
    is_blocked_by (X1,J2,[J1,J2],Scc),
    SumZ is Z1 +  Z2,
```

```
    K is SumZ mod 2 + 1,
    nth1(K,[J1,J2],Dictator),
    range(Scc,Image_of_Scc),
    member(C,Image_of_Scc),
    dictatorship(Dictator,Image_of_Scc,C).
```

付録４．シミュレーション事例集

● 遂行シミュレーション１

正準メカニズムの Prolog プログラムを用いてナッシュ均衡及びナッシュ遂行の可能性をシミュレーションによってテストする。同時に理論的に与えられているナッシュ遂行のための条件が満たされているかどうかも、具体的にチェックする。

```
/*  ナッシュ遂行のテスト */
test_impl(GF,Scc,Is,N):-
    trim_stacks,    ( \+ (scc(Scc,_,_)) -> (wn('no such Scc.'),fail);true),
    subset_of_agents(Is,N),   \+ member(N, [0,1]),
    open_output_file(Strm,Scc),  init_summary(Scc), clear_br_results,
    H = h0, display_scc(Scc,Is), display_scc(Strm,Scc,Is),  collect_statistics(Stat1),
%
    check_on_scc(GF,Is,Scc,H,Strm),    check_off_scc(GF,Is,Scc,H,Strm),
    collect_statistics(Stat2),  write_summary,  write_statistics(Stat1,Stat2),
    close_output_file(Strm),  write('save br_results? (yes)'),  read(Save),
    (Save = yes -> save_br_results(_,_);true),    wn('end'),    trim_stacks.
```

遂行シミュレーション２

社会選択関数の内外の結果について、それぞれナッシュ均衡をチェックするサブルーチン。ただし画面とファイルへの結果書き出しの部分については省略した。

対応内のパタンのチェック(check_on_scc/5)では各パタンに（正直）均衡が一つ見つかると先にすすむ。また対応外のパタン(check_off_scc/5)で均衡が検出されると以降メッセージ生成が打ち切られる。このため遂行不能の諸ケースでは、すべて１～２分のうちに完了している。なおルール 3 は逸脱の誘因チェック時だけに用いた。

```
 % check the outcomes on the domain of the scc.
check_on_scc(GF,Is,F,H,Strm):-
    setof((S,O), D^(state(S),alternative(O),scc(F,S,D),member(O,D)),  ON),
    forall(member((S,O),ON),  ((test_nash(GF,[O,S],[_P,F,H],_Msg,Is,yes) -> Yes=yes; Yes=no),
      update_summary([S,O,Yes,in]))),!.
 % check the outcomes off the domain of the scc.
check_off_scc(GF,Is,F,H,Strm):-
    setof((S,O),  D^(state(S),alternative(O),scc(F,S,D),\+member(O,D)),  OFF),
```

```
    forall(member((S,O),OFF),   ((test_nash(GF,[O,S],[_P,F,H],_M,Is,yes)  ->  No=yes; No=no),
       update_summary([S,O,No,out])  )),!.
```

- **遂行シミュレーション３**

```
  /*  ナッシュ均衡のテスト */
test_nash(GF,[C,S],[P,Scc,H],Msg,Is,Result):-
    nl,write('  Now checking m_profiles. Wait... '),nl,  member(GF,[gM,gD]),
  % P=3 (integer-game) may be excluded for simplicity since it never brings about NE.
    (GF = gM -> member(P,[1,2]);true),
    (GF = gD -> (member(P,[1,2,3]),N = 2);true),
    G =.. [GF,P,Scc],    subset_of_agents(Is,N),   E = environment([Is,_,_],[N,_,_],_),
    mtest_0(GF, Msg,Is),    nash(C,S,Msg,G,H,E,Result),!.
```

- **シミュレーション例１**

図にモデル１の選好領域においてSCC（ｆ）の遂行シミュレーション結果を示す。 [s2,z]はSCC外の均衡であり、遂行は失敗している。原因は ｆ が単調でないためである。実際、モデル４で示したプログラムを使えば、S2において ｆ から外れる ｚ に選好逆転が生じないことが分かる。なお、P1sは変異の処理ルール、Czsはその出力（可達集合－元案）、Lccは劣位集合である。ちなみに同モデルを一括方式で処理すると、メカニズム出力の枚挙・保存に数十秒かかるが、シミュレーション自体は数秒で終わる。

(・・途中省略・・)

- **シミュレーション例２**

SCC ｆ１ は単調だが本質的単調ではない。代替案 ｙ は状態 s1 で ｆ1 から外れるが、別の代替案wによる選好逆転が起き、単調性は保たれる。しかしwは ｆ の値域外である。

(‥途中省略‥)

- シミュレーション例３

本質的単調性を満たす SCC g1 の場合。

(・・途中省略・・・)

(・・途中省略・・・)

- **シミュレーション例４**

ｇ１をエージェント１と３の２人に絞った領域上での遂行可能性を、Danilov のメカニズムを用いてテストした。領域が変更されたので、ｇ１は本質的単調であるとは限らず、また仮にそうであっても非制限領域ではない２人の場合なので遂行できる保証はないことに注意する。この例ではたまたま本質的単調であり、かつ遂行できた。

Testing Nash implementability as for the members of this society: [1,3]

Scc: g1(s1) = [x,w] g1(s2) = [x,y]

is Maskin-monotone.

is Essentially-monotone.

checking the on-SCC patterns: [s1,w][s1,x][s2,x][s2,y]

For state s1,outcome=w,rule=1 message profile is

[w,y], 0

[w], 0

agent=1,P1s=[],Czs=[],Lcc=[w] <is_best_response>

agent=3,P1s=[1],Czs=[y],Lcc=[w,y] <is_best_response>

Br_Members=[1,3]

This action profile is a Nash equilibrium.

(・・途中省略・・・)

```
Summary
Result
[s1,w,yes,in]
[s1,x,yes,in]
[s1,y,no,out]
[s1,z,no,out]
[s2,w,no,out]
[s2,x,yes,in]
[s2,y,yes,in]
[s2,z,no,out]
```

- **Sccの生成とテスト**

所与の選好領域の Prolog モデルに対し、単調性や MR 特性など各種条件を満足する任意の Scc を自動生成できる。また計算量の問題はあるが、Scc を所与として、ナッシュ遂行可能なエージェントの選好を生成したり、あるいはナッシュ遂行可能な Scc と選好領域の補完的ペアの共変化も原理的には可能である。

```
% scc0: automatically generated SCC
generate_scc(scc0,[s1,X],[s2,Y]):-
    subset_of_alt(X,_N1),  X \= [], subset_of_alt(Y,_N2), Y \= [],
    clear_scc0,  % note: this position is sensitive.
    assert(scc(scc0,s1,X)),  assert(scc(scc0,s2,Y)).
clear_scc0:-  forall(scc(scc0,S,W),retract(scc(scc0,S,W))).
test_mrp(A,B,[J,K]):-
  generate_scc(scc0,A,B),has_MR_property(scc0,[J,K]).
test_ess(A,B,Is):-
  subset_of_agents(Is,_N),generate_scc(scc0,A,B),ess_monotone(scc0,Is).
```

- **Scc生成実験１**

２人と２代替案の非制限領域（その強選好部分）において、可能な SCC を枚挙した。以下にその実験データを表計算で整理し、パレート最適性を満たす SCC を抽出した結果を示す。

scc(s1)	scc(s2)	scc(s3)	scc(s4)	mm	em	mr	ir	bad	nvp	rvp	unan	po	co	dict	neli	mlib	mr+ir+em
[x]	[y]	[y]	[y]	1	1	0	1	0	0	1	1	1	0	0	0	0	0
[x]	[x]	[y]	[y]	1	1	1	1	0	0	1	1	1	0	1	0	0	1
[x]	[x,y]	[y]	[y]	1	1	0	1	0	0	1	1	1	0	0	0	1	0
[x]	[y]	[x]	[y]	1	1	1	1	0	0	1	1	1	0	1	0	0	1
[x]	[x]	[x]	[y]	1	1	0	1	0	0	1	1	1	0	0	0	0	0
[x]	[x,y]	[x]	[y]	1	1	0	1	0	0	1	1	1	0	0	0	1	0
[x]	[y]	[x,y]	[y]	1	1	0	1	0	0	1	1	1	0	0	0	1	0
[x]	[x]	[x,y]	[y]	1	1	0	1	0	0	1	1	1	0	0	0	1	0
[x]	[x,y]	[x,y]	[y]	1	1	0	1	0	1	1	1	1	0	0	0	1	0

- **Scc生成実験２**

３人と２代替案の非制限領域（その強選好部分）において、可能な SCC（非空のみ６５６１個）を枚挙した。以下にその実験データを表計算で整理し、本質的単調性を満たし、かつパレート最適性を満たす SCC１２９個を抽出した結果を示す。

経済学紀要第30集(1) 投稿原稿

scc(s1)	scc(s2)	scc(s3)	scc(s4)	scc(s5)	scc(s6)	scc(s7)	scc(s8)	mm	em	mr	ir	bad	,nvp	rnvp	unan	po	co	dict	neli	mlib	mr+ir+em
[b]	[b]	[b]	[b]	[b]	[b]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a]	[b]	[b]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a,b]	[b]	[b]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[b]	[b]	[a]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a]	[b]	[a]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a]	[b]	[a]	[b]	[a]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	1	1	0	0
[b]	[a,b]	[b]	[a]	[b]	[a]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a,b]	[b]	[a]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a,b]	[b]	[a]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[b]	[b]	[a,b]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a]	[b]	[a,b]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a]	[b]	[a,b]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[a,b]	[b]	[a,b]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a,b]	[b]	[a,b]	[b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[b]	[b]	[b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a]	[b]	[b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[a]	[a]	[b]	[b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	1	1	0	0
[b]	[b]	[a,b]	[a]	[b]	[b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a,b]	[b]	[b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[a,b]	[a,b]	[b]	[b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[b]	[b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a]	[b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[a]	[b]	[a]	[b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a]	[b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[b]	[a]	[a]	[b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a]	[a]	[a]	[b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[a]	[a]	[b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[a,b]	[a]	[b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[a]	[a,b]	[a]	[b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[a,b]	[a]	[b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[b]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	1	1	0	0
[b]	[b]	[b]	[a]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a]	[b]	[a]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[a]	[a]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a]	[a]	[a]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[a]	[a]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[a,b]	[a]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a]	[a,b]	[a]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[a,b]	[a]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a,b]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[a,b]	[a,b]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a,b]	[a]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[b]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[a]	[b]	[a]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[b]	[a]	[a]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a]	[a]	[a]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[a]	[a]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[a,b]	[a]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[a]	[a,b]	[a]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[a,b]	[a]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a,b]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a,b]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0

[b]	[b]	[a,b]	[a,b]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a,b]	[a,b]	[a]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[b]	[b]	[a,b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a]	[b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a]	[b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[a]	[a]	[b]	[a,b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[a]	[a]	[b]	[a,b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[a,b]	[a]	[b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[a,b]	[a,b]	[a]	[b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[a,b]	[b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a,b]	[b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[a,b]	[a,b]	[b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a,b]	[b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[b]	[a,b]	[a,b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[a]	[a,b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[b]	[a]	[a,b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[a]	[a]	[a,b]	[a,b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[a,b]	[a]	[a]	[a,b]	[a,b]	[a]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[a,b]	[a]	[a,b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a]	[a,b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[a,b]	[a,b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[b]	[a,b]	[a,b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[a,b]	[a,b]	[a,b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a,b]	[a,b]	[a,b]	[a]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[b]	[b]	[b]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a]	[b]	[b]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[a,b]	[a]	[b]	[b]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[a,b]	[b]	[b]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[a,b]	[a,b]	[b]	[b]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[b]	[b]	[a]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a]	[b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[a]	[b]	[a]	[b]	[a]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a]	[b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[b]	[a,b]	[a]	[b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a]	[a,b]	[a]	[b]	[a]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a]	[b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[a,b]	[b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0

[b]	[a,b]	[b]	[a,b]	[b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[a,b]	[a,b]	[b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a,b]	[b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[b]	[a,b]	[a]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[a]	[a,b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a]	[b]	[a]	[a,b]	[a]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[a,b]	[b]	[a]	[a,b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[a,b]	[a]	[a,b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a]	[a,b]	[a]	[a,b]	[a]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a]	[a,b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[a,b]	[a,b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[b]	[a,b]	[a,b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[a,b]	[a,b]	[a,b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a,b]	[a,b]	[a]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[b]	[b]	[a,b]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	0	0
[b]	[b]	[b]	[a]	[b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a]	[b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[a,b]	[a]	[b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a]	[b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[a,b]	[b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	0	0
[b]	[a,b]	[b]	[a,b]	[b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[a,b]	[a,b]	[b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a,b]	[b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[b]	[a,b]	[a,b]	[a,b]	[a]	1	1	0	0	0	0	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[a]	[a,b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[b]	[a]	[a,b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[a,b]	[a]	[a,b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a]	[a,b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[b]	[a,b]	[a,b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[b]	[a,b]	[a,b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[b]	[a,b]	[a,b]	[a,b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0
[b]	[a,b]	[a,b]	[a,b]	[a,b]	[a,b]	[a,b]	[a]	1	1	0	0	0	1	1	1	1	0	0	1	1	0

- **Scc生成実験 3**

モデル１の制限領域において、可能な SCC を枚挙し、本質的単調性を満たす SCC を抽出した。３人以上の制限領域では本質的単調性はナッシュ遂行の十分条件[9]であるから、これらはすべて遂行可能のはずである。

```
9 ?- results_to_file([extract_em,F,[1,2,3],C],'jp_em.txt',0).
(途中省略。以下の内容がファイルに保存される。)
extract_em([[s1,[z]],[s2,[z]]], [1,2,3], [mm,em,no,no,bad(w),nvp,rvp,unan,po,no,dict,neli,mlib]).
extract_em([[s1,[y]],[s2,[y]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,po,no,dict,neli,mlib]).
extract_em([[s1,[w]],[s2,[y]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,y]],[s2,[y]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[y,z]],[s2,[y,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,po,no,no,neli,mlib]).
extract_em([[s1,[w,z]],[s2,[y,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,y,z]],[s2,[y,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[x]],[s2,[x]]], [1,2,3], [mm,em,no,no,bad(w),nvp,rvp,unan,po,no,dict,neli,mlib]).
extract_em([[s1,[x,z]],[s2,[x,z]]], [1,2,3], [mm,em,no,no,bad(w),nvp,rvp,unan,po,no,no,neli,mlib]).
extract_em([[s1,[x,y]],[s2,[x,y]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,po,no,no,neli,mlib]).
extract_em([[s1,[w,x]],[s2,[x,y]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,x,y]],[s2,[x,y]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[x,y,z]],[s2,[x,y,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,po,no,no,neli,mlib]).
extract_em([[s1,[w,x,z]],[s2,[x,y,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,x,y,z]],[s2,[x,y,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w]],[s2,[w]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,z]],[s2,[w,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w]],[s2,[w,y]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,y]],[s2,[w,y]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,z]],[s2,[w,y,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,y,z]],[s2,[w,y,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,x]],[s2,[w,x]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,x,z]],[s2,[w,x,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,x]],[s2,[w,x,y]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,x,y]],[s2,[w,x,y]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,x,z]],[s2,[w,x,y,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
extract_em([[s1,[w,x,y,z]],[s2,[w,x,y,z]]], [1,2,3], [mm,em,no,no,no,nvp,rvp,unan,no,no,no,neli,mlib]).
end_of_run
```

文献

[1] Danilov, V. (1992). Implementation via Nash equilibria. Econometrica 60(1): 43-56.

[2] Dutta, B. and A. Sen(1991). A necessary and sufficient condition for two-person Nash implementation. Review of Economic Studies 58:121-8.
[3] Maskin, E. (1985). The theory of implementation in Nash equilibrium: a survey. In L.Hurwicz, D. Schmeidler, and H.Sonnenchein(eds.), Social Goals and Social Organization: Essays in Memory of Elisha Pozner, Cambridge University Press, pp.173-204.
[4] Maskin, E. (1999). Nash equilibrium and welfare optimality. Review of Economic Studies 66:23-38.
[5] Maskin, E. and T. Sjostrom(2002). Implementation theory. In K. Arrow, A. Sen, and K. Suzumura (eds.), Handbook of Social Choice and Welfare, Vol. 1, North-Holland, pp.237-288.
[6] Moore, J. and R. Repullo(1990). Nash implementation: A full charaterization. Econometrica 58(5):1083-99.
[7] Saijo, T. (1988). Strategy space reduction in Maskins theorem: Sufficient conditions for Nash implementation. Econometrica 56(3): 693-700.
[8] Wielemaker, J. (1994). SWI-Prolog 1.6 Reference Manual. In Kantrowitz(ed.), CMU AI Repository. Prime Time Freeware for AI, Issue 1-1. (software with manual is downloadable from URL http://www-2.cs.cmu.edu/afs/cs/project/ai-repository/ai/lang/prolog/impl/prolog/swi_pl/0.html)
[9] Yamato, T. (1992). On Nash implementation of social choice correspondences. Games and Economic Behavior 4:484-92.
[10] ロイド, J.W. (1987). 『論理プログラミングの基礎』佐藤雅彦・森下真一(訳), 産業図書.