PROCEDURI MAPLE PENTRU REŢELE DE CELULE CUPLATE CU ARCURI MULTIPLE

Mădălina Roxana Buneci, *Assoc. Professor, University Constantin Brâncuşi*

REZUMAT: *Un sistem cu celule cuplate este o reţea de sisteme dinamice (celule), reprezentat printr-un graf orientat ce indică modul în care celulele interacţionează şi care celule şi tipuri de interacţiuni sunt echivalente. Golubitsky şi Stewart [1-3] au introdus un cadru formal pentru a obţine informaţie despre dinamica sistemului folosind topologia grafului de interacţiuni. Scopul acestei lucrări este de a prezenta modul în care mediul de programare Maple poate fi utilizat pentru realizarea rapidă a unor proceduri Maple pentru implementarea acestui model.*

Cuvinte cheie: reţea celulară, procedură Maple, graf orientat

1. Proceduri Maple pentru grafuri orientate etichetate

A *graf orientat etichetat* orientat G constă în:

1. O mulţime V_G de vârfuri sau noduri.
2. A mulţime E_G de muchii orientate sau arce.
3. Două aplicaţii H: $E_G \to V_G$ (H(e) extremitatea finală a lui e) şi T: $E_G \to V_G$ (T(e) extremitatea iniţială a lui e).
4. O relaţie de echivalenţă $\sim_V$ peV_G.
5. O relaţie de echivalenţă $\sim_E$ pe E_G.

Vom desemna un graf etichetat orientat prin $G(V_G, E_G, T, H, \sim_V, \sim_E)$. Dacă e $\in E_G$ şi T(e)=u, H(e)=v, atunci vom folosi notaţia e=(v,u). Subliniem că o muchia orientată e nu este determinată în mod necesar de u şi v. În definiţia precedentă, muchiile orientate multiple sunt permise (adică este posibil să avem $H(e_2)$ şi $T(e_1) = T(e_2)$ dar $e1 \neq e2$), precum şi auto-interacţiunile (adică, este permis ca H(e) = T (e)). Golubitsky şi

MAPLE PROCEDURES FOR COUPLED CELL NETWORKS WITH MULTIPLE ARROWS

Mădălina Roxana Buneci, *Assoc. Professor, University Constantin Brâncuşi*

ABSTRACT: *A coupled cell system is a network of dynamical systems (cells), represented by a directed graph that indicates how cells are coupled and which cells are equivalent. Golubitsky and Stewart [1-3] introduced a formal framework for obtaining information about system dynamics from the topology of the graph of couplings. The purpose of this paper is to present how Maple environment can be used for rapid development of procedures implementing this model (multiple arrows and self-couplings are allowed).*

Keywords: cell networks, Maple procedure, directed graph

1. Maple procedures for labeled directed graphs with

A *labeled directed graph* G consists of the following:

6. A set V_G of vertices or nodes.
7. A set E_G of directed edges or arcs.
8. Two maps H: $E_G \to V_G$ (H(e) head or target of e) and T: $E_G \to V_G$ (T(e) tail or source of e).
9. An equivalence relation $\sim_V$ on vertices in V_G.
10. An equivalence relation $\sim_E$ on edges in E_G.

We shall denote a labeled directed graph by $G = G(V_G, E_G, H, T, \sim_V, \sim_E)$. If e $\in E_G$ and T(e)=u, H(e)=v, then we shall use the notation e=(v,u). Notice that the directed edge e is not necessarily determined by u and v. In the previous definition multiple arrows are allowed (it is possible to have $H(e_1) = H(e_2)$ and $T(e_1) = T(e_2)$ for $e1 \neq e2$), as well as self-coupling is allowed (that is, we allow

Stewart [1,2] au utilizat în definiţia reţelelor de celule cuplate următoarea condiţie de consistenţă: “muchiile orientate echivalente au extremităţile iniţiale, respectiv extremităţile finale echivalente”. Nu vom impune această restricţie.

Vom reprezenta în Maple un graf orientat etichetat $G(V_G, E_G, T, H, \sim_V, \sim_E)$ prin două liste (presupunând că V_G şi E_G sunt mulţimi finite): una pentru vârfuri, şi alta pentru muchiile orientate. Vom asocia fiecărei clase de echivalenţă o culoare ce poate fi exprimată în Maple în modelul RGB/HUE sau specificată printr-un identificator predefinit. Fiecare vârf va fi reprezentat ca o listă cu două elemente: un identificator (nume) al vârfului şi culoarea corespunzătoare clasei de echivalenţă a vârfului. Fiecare muchie orientată va fi reprezentată printr-o listă cu două elemente: primul element este dat de lista a două vârfuri (extremitatea iniţială şi extremitatea finală), iar cel de al doilea de culoarea corespunzătoare clasei de echivalenţă a muchiei.

De exemplu, să considerăm graful orientat etichetat G for pentru care V_G = {1, 2, 3, 4, 5}, clasele de echivalenţă pe V_G sunt {1}, {2,3} şi {4,5}, E_G ={e_1=(1,1), e_2= (1,2), e_3= (1,2), e_4= (1,3), e_5=(1,3), e_6= (2,4), e_7= (2,4), e_8= (2,5), e_9= (2,5), e_{10}= (3,4), e_{11}= (3,5), e_{12}= (4,1), e_{13}= (5,1)}, clasele de echivalenţă pe E_G sunt{e_1, e_2, e_3, e_4, e_5}, {e_6, e_8, e_{10}}, {e_7, e_9, e_{12}} şi {e_{13}} (să observăm existenţa muchiilor multiple şi a auto-interacţiunii e_1=(1,1)). Informaţia referitoare la acest graf este conţinută în următoarele două liste Maple:

```
>V1:=[[1,COLOR(RGB,1,1,0)],[2,
COLOR(RGB,1,0,1)],[3,COLOR(RGB
,1,0,1)],[4,COLOR(RGB,0,1,1)],
[5,COLOR(RGB,0,1,1)]];
>
E1:=[[[1,1],COLOR(RGB,1,0,0)],
[[1,2],COLOR(RGB,1,0,0)][[1,2]
,
COLOR(RGB,1,0,0)],[[1,3],COLOR
(RGB,1,0,0)],[[1,3],COLOR(RGB,
1,0,0)],[[2,4],COLOR(RGB,0,0,1
)],[[2,4],COLOR(RGB,1,1/2,2/3)
```

H(e) = T (e)). Golubitsky and Stewart [1,2] used in the definition of the coupled cell networks (that permits multiple arrows and self-couplings) the following consistency condition “equivalent directed edges have equivalent tails and heads”. We shall not work under this assumption.

We shall represent in Maple a labeled directed graph $G(V_G, E_G, H, T, \sim_V, \sim_E)$ using two lists: one for vertices and the other for edges. We shall associate to each equivalence class a color expressed in Maple in the RGB/HUE model or specified by predefined names. Each vertex will be represented as a list of two elements: the identifier (name) of the vertex and the color corresponding to its equivalence class. Each directed edge will be also represented as a list with two components: a list of two vertices (the tail and the head of the edge) and the color corresponding to the equivalence class of the edge.

For instance, let us consider the graph $G(V_G, E_G, H, T, \sim_V, \sim_E)$ for which V_G = {1, 2, 3, 4, 5}, the equivalence classes on V_G are {1}, {2,3} and {4,5}, E_G ={e_1=(1,1), e_2= (1,2), e_3= (1,2), e_4= (1,3), e_5=(1,3), e_6= (2,4), e_7= (2,4), e_8= (2,5), e_9= (2,5), e_{10}= (3,4), e_{11}= (3,5), e_{12}= (4,1), e_{13}= (5,1)}, the equivalence classes on E_G are {e_1, e_2, e_3, e_4, e_5}, {e_6, e_8, e_{10}}, {e_7, e_9, e_{12}} and {e_{13}} (let us notice the existence of multiple arrows and the self-couplings e_1 = (1,1)). The information concerning this graph is contained in following two Maple lists:

```
>V1:=[[1,COLOR(RGB,1,1,0)],[2,
COLOR(RGB,1,0,1)],[3,COLOR(RGB
,1,0,1)],[4,COLOR(RGB,0,1,1)],
[5,COLOR(RGB,0,1,1)]];
>
E1:=[[[1,1],COLOR(RGB,1,0,0)],
[[1,2],COLOR(RGB,1,0,0)][[1,2]
,
COLOR(RGB,1,0,0)],[[1,3],COLOR
(RGB,1,0,0)],[[1,3],COLOR(RGB,
1,0,0)],[[2,4],COLOR(RGB,0,0,1
)],[[2,4],COLOR(RGB,1,1/2,2/3)
],[[2,5],COLOR(RGB,0,0,1)],[[2
,5],COLOR(RGB,1,1/2,2/3)],[[3,
4],COLOR(RGB,0,0,1)],[[3,5],CO
```

```
],[[2,5],COLOR(RGB,0,0,1)],[[2
,5],COLOR(RGB,1,1/2,2/3)],[[3,
4],COLOR(RGB,0,0,1)],[[3,5],CO
LOR(RGB,0,0,1)],[[5,1],COLOR(R
GB,0,1,0)],[[4,1],COLOR(RGB,1,
1/2,2/3)]];
```

Proceduri Maple pentru determinarea culorii unui vârf, culorii tuturor vârfurilor sau muchiilor, întreaga clasă a unui vârf, culorile muchiilor (posibil multiple) sunt uşor de implementat. În cazul în care muchiile multiple şi autointeracţiunile nu sunt permise, se pot folosi comenzile din pachetul **GraphTheory** pentru reprezentarea grafică a grafului can (vârfurile şi muchiile pot fi desenate cu culorile specificate în listele corespunzătoate)

Dându-se o listă EE de muchii orientate, procedura **in_vertices(EE,u)** calculează lista aşa numiţilor „in-neighbors” ai vârfului u (extremităţile iniţiale ale muchiilor având u ca extremitatea finală).

```
> in_vertices:=proc(EE,u)
local i, L;
 L:=[];
 for i from 1 to nops(EE) do if
EE[i][2]=u then
L:=[op(L),EE[i][1]] end if;
 end do;
RETURN(L)
end proc:
```

Dându-se o listă E de muchii orientate însoţite de culorile asociate claselor de echivalenţă, procedura **in_colored_edges(E,u)** calculează lista aşa numitelor muchii input pentru u (extremitatea finală a fiecărei astfel de muchie este u). Pentru fiecare muchie este memorată şi clasă de echivalenţă desenată prin culoare.

```
> in_colored_edges:=proc(E,u)
local i, L;
```

```
LOR(RGB,0,0,1)],[[5,1],COLOR(R
GB,0,1,0)],[[4,1],COLOR(RGB,1,
1/2,2/3)]];
```

Maple procedures for determined the color of a vertex, all colors of the vertices (respectively, directed edges), the entire class of a vertex, the colors of a (possible multiple) directed edge are very easy to implement. If we do not allow multiple arrows and self-couplings, the Maple commands from the package **GraphTheory** can be used to draw the graph (the vertices and the edges can be painted with the colors specified in the corresponding lists).

Given a list of directed edges EE, the procedure **in_vertices(EE,u)** returns a list of in-neighbors of the vertex u (the tails of the edges having u as head).

```
> in_vertices:=proc(EE,u)
local i, L;
 L:=[];
 for i from 1 to nops(EE) do if
EE[i][2]=u then
L:=[op(L),EE[i][1]] end if;
 end do;
RETURN(L)
end proc:
```

Given a list of directed edges E with colors, the procedure **in_colored_edges(E,u)** returns a list of so called input edges of u (the head of each such edge is u). For each edge the color is also stored.

```
> in_colored_edges:=proc(E,u)
local i, L;
L:=[];
for i from 1 to nops(E) do if
E[i][1][2]=u then
L:=[op(L),E[i]] end if;
 end do;
RETURN(L)
end proc:
```

2. Maple procedures for studying the in-groupoid associated to a labeled

```
L:=[];
for i from 1 to nops(E) do if
E[i][1][2]=u then
L:=[op(L),E[i]] end if;
 end do;
RETURN(L)
end proc:
```

2. Proceduri Maple pentru studiul in-grupoidului asociat unui graf orientat etichetat

Cadrul formal introdus de Golubitsky şi Stewart pentru a obţine informaţie despre dinamica sistemului de celule cuplate folosind doar topologia grafului de interacţiuni (descris în detaliu în [3]) se bazează pe noţiunea de grupoid. Un grupoid este asemănător unui grup cu excepţia faptului că multiplicarea este doar parţial definită. Mai precis, un grupoid este o mulţime G, împreună cu o submulţime $G^{(2)} \subset G\times G$, şi două aplicaţii: o aplicaţie produs

$$G^{(2)} \ni (x,y) \mapsto xy \in G$$

şi o aplicaţie de inversare

$$G \ni x \mapsto x^{-1} \in G$$

astfel încât să fie îndeplinite următoarele condiţii:

(1) $\left(x^{-1}\right)^{-1} = x$

(2) If $(x,y)\in G^{(2)}$ şi $(y,z)\in G^{(2)}$, atunci (xy, z), $(x,yz) \in G^{(2)}$ şi $(xy)z = x(yz)$

(3) $(x, x^{-1})\in G^{(2)}$, şi dacă $(y, x)\in G^{(2)}$, atunci $(yx)x^{-1} = y$.

(4) $(x^{-1}, x)\in G^{(2)}$, şi dacă $(x, y)\in G^{(2)}$, atunci $x^{-1}(xy) = y$.

Aplicaţiile r şi d pe G, definite prin formulele $r(x) = xx^{-1}$ şi $d(x) = x^{-1}x$, sunt numite aplicaţia ţintă şi aplicaţia sursă. Rezultă uşor din definiţie că ele au o imagine comună numită spaţiul unităţilor grupoidului şi notată $G^{(0)}$. Fibrele aplicaţiilor r şi d se notează cu $G^u = r^{-1}(\{u\})$ şi respectiv $G_u = d^{-1}(\{u\})$. Pentru u şi v ân $G^{(0)}$, (r,d)-fibra se notează cu $G_v^u = G^u \cap G_v$. Este uşor de observat că G_u^u este un grup, numit *grupul de izotropie* corespunzător lui u.

directed graph

The formal framework introduced by Golubitsky and Stewart for obtaining information about system dynamics from the topology of the graph of couplings (extensively described in [3]) is based on the notion of groupoid. A groupoid is like a group with multiplication only partially defined. More precisely, a groupoid is a set G, together with a distinguished subset $G^{(2)} \subset G\times G$, and two maps: a product map

$$G^{(2)} \ni (x,y) \mapsto xy \in G$$

and an inverse map

$$G \ni x \mapsto x^{-1} \in G$$

such that the following relations are satisfied:

(1) $\left(x^{-1}\right)^{-1} = x$

(2) If $(x,y)\in G^{(2)}$ and $(y,z)\in G^{(2)}$, then (xy, z), $(x,yz) \in G^{(2)}$ and $(xy)z = x(yz)$

(3) $(x, x^{-1})\in G^{(2)}$, and if $(y, x)\in G^{(2)}$, then $(yx)x^{-1} = y$.

(4) $(x^{-1}, x)\in G^{(2)}$, and if $(x, y)\in G^{(2)}$, then $x^{-1}(xy) = y$.

The maps r and d on G, defined by the formulae $r(x) = xx^{-1}$ and $d(x) = x^{-1}x$, are called the *range* and the *source* (*domain*) maps. It follows easily from the definition that they have a common image called the unit space of G, which is denoted $G^{(0)}$. The fibres of the range and the source maps are denoted $G^u = r^{-1}(\{u\})$ and $G_u = d^{-1}(\{u\})$, respectively. For u and v in $G^{(0)}$, (r,d)-fibre is $G_v^u = G^u \cap G_v$. It is easy to see that G_u^u is a group, called the *isotropy group* at u.

The relation $u \sim v$ if and only if $G_v^u \neq \varnothing$ is an equivalence relation on $G^{(0)}$. Its equivalence classes are called *orbits* and the orbit of a unit u is denoted [u]. A groupoid is called *transitive* if and only if it has a single orbit.

Let us recall the construction of the in-groupoid (symmetry groupoid) associated to a labeled directed graph (a slightly modified version of [2]).

Let $G = G(V_G, E_G, H, T, \sim_V, \sim_E)$ be a

Relaţia u ~ v dacă şi numai dacă $G_v^u \neq \emptyset$ este o relaţie de echivalenţă pe $G^{(0)}$. Claseele ei de echivalentă se numesc *orbite* iar orbit unei unităţi u se notează cu [u]. Un grupoid se numeşte tranzitiv dacă şi numai da că are o singură orbită .

Reamintim contrucţia aşa numitului in-grupoid (grupoidul de simetrie) associat unui graf etichetat orientat (o variantă uţor modificată faţă de cea din [2]).

Fie $G(V_G, E_G, T, H, \sim_V, \sim_E)$ un graf orientat etichetat. Pentru fiecare $u \in V_G$ notăm

$G(\_, u) = \{e \in E_G: H(e)=u\} \cup \{u\}$ (muchii input pentru u la care adăugăm u)

Fie două vârfuri echivalente u şi v ($u \sim_V v$). Dacă există o bijecţie γ de $G(\_, v)$ la $G(\_, u)$ astfel încât $\gamma(v) = u$, iar pentru orice $e \in G(\_, v)$ avem

$$T(e) \sim_V T(\gamma(e))$$

şi

$$e \sim_E \gamma(e),$$

atunci γ este numit *izomorfism input* de la v la u.

Fie

$$I_v^u = \{ (\gamma, u, v): \gamma \text{ este izomorfism input de la v la u } \}$$

şi

$$I(G) = \bigcup_{(u,v)\in V_G\times V_G} I_v^u$$

Definim o operaţie de multiplicare pe I(G) în felul următor. Elementele (γ_1, u_1, v_1) şi (γ_2, u_2, v_2) pot fi multiplicate doar dacă $v_1 = u_2$, şi în acest caz

$$(\gamma_1, u_1, v_1)\,(\gamma_2, u_2, v_2) = (\gamma_1 \circ \gamma_2, u_1, v_2),$$

unde $\gamma_1 \circ \gamma_2$ desemnează compunerea uzuală a funcţiilor γ_1 şi γ_2. Este uşor de verificat că dacă γ_1 este un izomorfism input de la u_2 la u_1 şi dacă γ_2 este un izomorfism input de la v_2 la u_2, atunci $\gamma_1 \circ \gamma_2$ este un izomorfism input de la v_2 la u_1. For fiecare (γ, u, v) definim inversul prin $(\gamma, u, v)^{-1} = (\gamma^{-1}, v, u)$. Mulţimea I(G) înzestrată cu multiplicarea parţială şi aplicaţia de inversare definite mai sus este un grupoid numit *in-grupoidul* associat grafului orientat etichetat $G(V_G, E_G, T, H, \sim_V, \sim_E)$.

labeled directed graph. For each $u \in V_G$ let us denote

$G(\_, u) = \{e \in E_G: H(e)=u\} \cup \{u\}$ (input edges of u and u)

Let us consider two equivalent vertices u and v ($u \sim_V v$). If there is a bijection γ from $G(\_, v)$ to $G(\_, u)$ such that $\gamma(v) = u$ and for all $e \in G(\_, v)$ we have

$$T(e) \sim_V T(\gamma(e))$$

and

$$e \sim_E \gamma(e),$$

then γ is called an *input isomorphism* from v to u.

Let us define

$$I_v^u = \{ (\gamma, u, v): \gamma \text{ is an input isomorphism from v to u } \}$$

and

$$I(G) = \bigcup_{(u,v)\in V_G\times V_G} I_v^u$$

We define a product operation on I(G) in the following way. The elements (γ_1, u_1, v_1) and (γ_2, u_2, v_2) can be multiplied only when $v_1 = u_2$, and in this case

$$(\gamma_1, u_1, v_1)\,(\gamma_2, u_2, v_2) = (\gamma_1 \circ \gamma_2, u_1, v_2),$$

where $\gamma_1 \circ \gamma_2$ denotes the usual composition of the function γ_1 and γ_2. It is easy to check that if γ_1 is an input isomorphism from u_2 to u_1 and if γ_2 is an input isomorphism from v_2 to u_2, then $\gamma_1 \circ \gamma_2$ is an input isomorphism from v_2 to u_1. For each (γ, u, v) we define its inverse by $(\gamma, u, v)^{-1} = (\gamma^{-1}, v, u)$. The set I(G) endowed with the product and inverse maps defined above can be viewed as groupoid, which will be called *in-groupoid* associated to the labeled directed graph $G = G(V_G, E_G, H, T, \sim_V, \sim_E)$.

Let $G = G(V_G, E_G, H, T, \sim_V, \sim_E)$ be a labeled directed graph. Let us assume that V_G and E_G are finite sets. In order to obtain the elements of the groupoid I(G), for each two vertices u and v with the same color, we generate the bijections between $G(\_, u)$ and $G(\_, v)$ (or equivalently, the permutations of {1, …,n} where n is the cardinality of $G(\_, u) \setminus \{u\}$) and we check which of them are compatible with the colors (of vertices and edges). Those are elements of I(G). The

Fie G(V_G, E_G, T, H, ~V, ~E) un graf orientat etichetat Presupunem că V_G şi E_G sunt mulţimi finite. Pentru a obţine elementele grupoidului I(G), pentru fiecare două vârfuri u şi v de aceeaşi culoare, generăm bijecţiile dintre G(_, u) şi G(_, v) (sau echivalent, permutările multimii {1, …,n} where n is cardinalul mulţimii G(_, u) \{u}) şi verificăm compatibilitatea lor cu culorile (vârfurilor şi muchiilor). Cele compatibile sunt elemente ale lui I(G). Procedura

in_groupoid_fibre(V,E,v,u)

calulează (r,d)-fibra lui I(G) corespunzând funcţiei identitate pe G(_, u) şi funcţiei identitate pe G(_, v). Parametrii ei sunt lista vârfurilor însoţite de culori V, lista muchiilor orientate însoţite de culori E, vârfurile u şi v. Utilizăm comanda Maple **permute** pentru a genera permutările, de aceea este necesar pachetul **combinat**. Un izomorfism input este representat ca o lista (prin identificarea lu cu o permutare compatibilă cu culorile). 0 este aşezat la începutul listei corespunzătoare oricărui izomofism input de la u la v pentru a preciza asocierea lui u cu v.

```
> with(combinat):
>
in_groupoid_fibre:=proc(V,E,v,
u)
 local
i,j,perm,p,L,test,Iu,Iv,EE;
 L:=[];
 if
vertex_color(V,u)<>vertex_colo
r(V,v) then RETURN(L) end if;
 Iu:=in_colored_edges(E,u);
 Iv:=in_colored_edges(E,v);
 if nops(Iu)<>nops(Iv) then
RETURN(L) end if;
 if nops(Iu)=0 then L:=[[0]];
RETURN(L) end if;
 perm:=permute(nops(Iu));
 for j from 1 to nops(perm) do
 p:=perm[j];
 i:=1; test:=1;
 while (i<=nops(Iu)) do
 if (Iu[i][2]=Iv[p[i]][2])and
```

procedure

in_groupoid_fibre(V,E,v,u)

computes the (r,d)-fibre of I(G) corresponding to the identity function on G(_, u) and the identity function on G(_, v). Its parameters are the list of vertices V (with colors), the list of directed edges (with colors) and the vertices u and v. We use the Maple command **permute** in order to generate the permutations, therefore we need the package **combinat**. The input isomorphisms are represented as list (identified them with permutations of compatible with the colors). 0 at the beginning of each list corresponding to an input isomorphism from u to v means that u is mapped in v.

```
> with(combinat):
>
in_groupoid_fibre:=proc(V,E,v,
u)
 local
i,j,perm,p,L,test,Iu,Iv,EE;
 L:=[];
 if
vertex_color(V,u)<>vertex_colo
r(V,v) then RETURN(L) end if;
 Iu:=in_colored_edges(E,u);
 Iv:=in_colored_edges(E,v);
 if nops(Iu)<>nops(Iv) then
RETURN(L) end if;
 if nops(Iu)=0 then L:=[[0]];
RETURN(L) end if;
 perm:=permute(nops(Iu));
 for j from 1 to nops(perm) do
 p:=perm[j];
 i:=1; test:=1;
 while (i<=nops(Iu)) do
 if (Iu[i][2]=Iv[p[i]][2])and

 (vertex_color(V,Iu[i][1][2])=v
 ertex_color(V,Iv[p[i]][1][2]))
then  i:=i+1;  else  test:=0;
i:=nops(Iu)+1
end if
end do;
if test=1 then
L:=[op(L),[0,op(p)]]; end if;
end do;
```

```
 (vertex_color(V,Iu[i][1][2])=v
 ertex_color(V,Iv[p[i]][1][2]))
then  i:=i+1;  else  test:=0;
i:=nops(Iu)+1
end if
end do;
if test=1 then
L:=[op(L),[0,op(p)]]; end if;
end do;
RETURN(L)
end proc:
> in_groupoid_fibre(V1,E1,5,4);
       [[0, 1, 2, 3], [0, 3, 2, 1]]
```

Procedura **show_groupoid_fibre(V,E,v,u)** afişează mai intuitiv lista bijecţiilor din fibra grupoidului, arătând asocierea lui u cu v şi asocierile dintre extremităţile iniţiale ale muchiilor din G(_, u) \{u} cu extremităţile iniţiale ale muchiilor din G(_, v) \{v}.

```
>
show_in_groupoid_fibre:=proc(V
,E,v,u)
local i,morph,Iu,Iv,L;

Iu:=[u,op(in_vertices([seq(E[i
][1],i=1..nops(E))],u))];
Iv:=[v,op(in_vertices([seq(E[i
][1],i=1..nops(E))],v))];
 L:=in_groupoid_fibre(V,E,v,u);
 for i from 1 to nops(L) do
morph:=[cat(convert(u,name),`-
>`,convert(v,name),`;
`,seq(cat(convert(Iu[j],name),
`-
>`,convert(Iv[L[i][j]],name),`
; `),j=2..nops(L[i])))];
 print(morph);
 end do
 end proc;
>
show_in_groupoid_fibre(V1,E1,5
,4);
       [4->5; 2->5; 2->2; 3->2; ]
       [4->5; 2->2; 2->2; 3->5; ]
RETURN(L)
end proc:
> in_groupoid_fibre(V1,E1,5,4);
       [[0, 1, 2, 3], [0, 3, 2, 1]]
```

The procedure **show_groupoid_fibre(V,E,v,u)** prints more intuitively the list of bijections in a groupoid fibre, showing the association of u with v and the associations of the tails of the edges in G(_, u) \{u} with the tails of the edges in G(_, v) \{v}.

```
>
show_in_groupoid_fibre:=proc(V
,E,v,u)
local i,morph,Iu,Iv,L;

Iu:=[u,op(in_vertices([seq(E[i
][1],i=1..nops(E))],u))];
Iv:=[v,op(in_vertices([seq(E[i
][1],i=1..nops(E))],v))];
 L:=in_groupoid_fibre(V,E,v,u);
 for i from 1 to nops(L) do
morph:=[cat(convert(u,name),`-
>`,convert(v,name),`;
`,seq(cat(convert(Iu[j],name),
`-
>`,convert(Iv[L[i][j]],name),`
; `),j=2..nops(L[i])))];
 print(morph);
 end do
 end proc;
>
show_in_groupoid_fibre(V1,E1,5
,4);
       [4->5; 2->5; 2->2; 3->2; ]
       [4->5; 2->2; 2->2; 3->5; ]
```

The procedure **show_in_groupoid(V,E,c)** prints the lists of bijections in all groupoid fibres. Its parameters are the list of vertices V (with colors), the list of directed edges (with colors) E and a name (or string) that will be use to denote the groupoid.

```
> show_in_groupoid:=proc(V,E,c)
 local VV,i,j;
```

Procedura **show_in_groupoid(V,E,c)** afişează listele bijecţiilor din toate fibrele grupoidului. Parametrii sunt lista vârfurilor însoţite de culori V, lista muchiilor orientate însoţite de culori E şi un nume ce va fi utilizat pentru a desemna grupoidul.

```
> show_in_groupoid:=proc(V,E,c)
 local VV,i,j;

VV:=[seq(V[i][1],i=1..nops(V))
];
 for i from 1 to nops(V1) do
 for j from 1 to nops(V1) do
 if
nops(in_groupoid_fibre(V,E,VV[
i],VV[j]))>0 then

print(cat(convert(c,name),`(`,
convert(VV[i],name),`, `,
convert(VV[j],name),`)`));
show_in_groupoid_fibre(V,E,VV[
i],VV[j])
end if;
 end do
 end do
 end proc:

> show_in_groupoid(V1,E1,I);
                  I(1, 1)
        [ 1->1; 1->1; 5->1; 4->5; ]
                  I(2, 2)
           [ 2->2; 1->2; 1->1; ]
           [ 2->2; 1->1; 1->2; ]
                  I(2, 3)
           [ 3->2; 1->2; 1->1; ]
           [ 3->2; 1->1; 1->2; ]
                  I(3, 2)
           [ 2->3; 1->3; 1->1; ]
           [ 2->3; 1->1; 1->3; ]
                  I(3, 3)
           [ 3->3; 1->3; 1->1; ]
           [ 3->3; 1->1; 1->3; ]
                  I(4, 4)
        [ 4->4; 2->4; 2->2; 3->2; ]
        [ 4->4; 2->2; 2->2; 3->4; ]
                  I(4, 5)
        [ 5->4; 2->4; 2->2; 3->2; ]
        [ 5->4; 2->2; 2->2; 3->4; ]
```

```
VV:=[seq(V[i][1],i=1..nops(V))
];
 for i from 1 to nops(V1) do
 for j from 1 to nops(V1) do
 if
nops(in_groupoid_fibre(V,E,VV[
i],VV[j]))>0 then

print(cat(convert(c,name),`(`,
convert(VV[i],name),`, `,
convert(VV[j],name),`)`));
show_in_groupoid_fibre(V,E,VV[
i],VV[j])
end if;
 end do
 end do
 end proc:

> show_in_groupoid(V1,E1,I);
                  I(1, 1)
        [ 1->1; 1->1; 5->1; 4->5; ]
                  I(2, 2)
           [ 2->2; 1->2; 1->1; ]
           [ 2->2; 1->1; 1->2; ]
                  I(2, 3)
           [ 3->2; 1->2; 1->1; ]
           [ 3->2; 1->1; 1->2; ]
                  I(3, 2)
           [ 2->3; 1->3; 1->1; ]
           [ 2->3; 1->1; 1->3; ]
                  I(3, 3)
           [ 3->3; 1->3; 1->1; ]
           [ 3->3; 1->1; 1->3; ]
                  I(4, 4)
        [ 4->4; 2->4; 2->2; 3->2; ]
        [ 4->4; 2->2; 2->2; 3->4; ]
                  I(4, 5)
        [ 5->4; 2->4; 2->2; 3->2; ]
        [ 5->4; 2->2; 2->2; 3->4; ]
                  I(5, 4)
        [ 4->5; 2->5; 2->2; 3->2; ]
        [ 4->5; 2->2; 2->2; 3->5; ]
                  I(5, 5)
        [ 5->5; 2->5; 2->2; 3->2; ]
        [ 5->5; 2->2; 2->2; 3->5; ]
```

The procedure **in_orbit(V,E,u)** computes the orbit in I(G) corresponding to the identity function on G(_, u).

I(5, 4)

[4->5; 2->5; 2->2; 3->2;]

[4->5; 2->2; 2->2; 3->5;]

I(5, 5)

[5->5; 2->5; 2->2; 3->2;]

[5->5; 2->2; 2->2; 3->5;]

Procedura **in_orbit(V,E,u)** calculează orbita lui I(G) corespunzătoare funcției identitate pe G(_, u).

```
> in_orbit:=proc(V,E,vv)
local i,L,VV;
L:=[];
VV:=[seq(V[i][1],i=1..nops(V))
];
for i from 1 to nops(VV) do
if
nops(in_groupoid_fibre(V,E,vv,
VV[i]))>0 then
L:=[op(L),VV[i]] end if;
end do;
RETURN(L)
end proc:

> in_orbit(V1,E1,4);
                [4, 5]
```

Procedura **in_orbits(V,E)** calculează toate orbitele lui I(G).

```
> in_orbits:=proc(V,E)
local i,L,Lo,vv,VV;
 L:=[];

VV:=[seq(V[i][1],i=1..nops(V))
];
 while(nops(VV)>0) do
 vv:=VV[1];Lo:=[VV[1]];
VV:=subsop(1=NULL,VV);
i:=1;
while (i<=nops(VV)) do
if
nops(in_groupoid_fibre(V,E,vv,
VV[i]))>0 then
Lo:=[op(Lo),VV[i]];VV:=subsop(
i=NULL,VV); else i:=i+1 end if
end do;
L:=[op(L),Lo];
end do;
 RETURN(L)
 end proc:
```

```
> in_orbit:=proc(V,E,vv)
local i,L,VV;
L:=[];
VV:=[seq(V[i][1],i=1..nops(V))
];
for i from 1 to nops(VV) do
if
nops(in_groupoid_fibre(V,E,vv,
VV[i]))>0 then
L:=[op(L),VV[i]] end if;
end do;
RETURN(L)
end proc:

> in_orbit(V1,E1,4);
                [4, 5]
```

The procedure **in_orbits(V,E)** computes the orbits of I(G).

```
> in_orbits:=proc(V,E)
local i,L,Lo,vv,VV;
 L:=[];

VV:=[seq(V[i][1],i=1..nops(V))
];
 while(nops(VV)>0) do
 vv:=VV[1];Lo:=[VV[1]];
VV:=subsop(1=NULL,VV);
i:=1;
while (i<=nops(VV)) do
if
nops(in_groupoid_fibre(V,E,vv,
VV[i]))>0 then
Lo:=[op(Lo),VV[i]];VV:=subsop(
i=NULL,VV); else i:=i+1 end if
end do;
L:=[op(L),Lo];
end do;
 RETURN(L)
 end proc:

> in_orbits(V1,E1);
          [[1], [2, 3], [4, 5]]
```

Let us notice that the elements of each groupoid G are determined by the elements of the groups G_w^w and by sets of the form $H=\{x_w \in G$, such that $r(x_w)=w\}$ such that $d(H)=[w]$, where w runs in a set U that intersects each orbit of G. Therefore a better

```
> in_orbits(V1,E1);
        [[1], [2, 3], [4, 5]]
```

Să facem observaţia că elementele fiecărui grupoid G sunt determinate de elementel grupurilor G_w^w şi de mulţimile de forma $H=\{x_w \in G,$ such that $r(x_w)=w\}$ cu proprietatea că $d(H)=[w]$, unde w parcurge o mulţime U ce intersectează fiecare orbită a lui G. De aceea o reprezentare mai eficientă a in-grupoidul asociat unui graf orientat etichetat se obţine alegând un reprezentant $w_{[u]}$ în fiecare orbită [u] şi stocând toate izomorfismele input de la $w_{[u]}$ la $w_{[u]}$. Pentru orice alt $v \in [u]$ este suficient să memorăm un izomorfism input de la v la $w_{[u]}$. O procedura Maple pentru construirea unei astfel de structuri poate fi uşor scrisă utilizând procedurile precedente. De asemenea având o astfel de structură, se pot dezvolta proceduri Maple pentru calculul tuturor elementelor in-grupoidului, determinarea orbitelor, etc. Dezvoltarea unui pachet complet de proceduri va fi făcută împreună cu studenţii de la Ingineria Sistemelor.

REFERINŢE

[1] M. Golubitsky, I. Stewart şi M. Pivato, *Symmetry Grupoids şi Patterns of Synchrony in Coupled Cell Networks*, Siam J. Applied Dynamical Systems, 2 (4) (2003), 609-646.

[2] M. Golubitsky, I. Stewart şi A. Torok, *Patterns of Synchrony in Coupled Cell Networks with Multiple Arrows*, Siam J. Applied Dynamical Systems, 4 (1) (2005), 78-100.

[3] M. Golubitsky şi I. Stewart, *Nonlinear dynamics of networks: the grupoid formalism*, Bulletin of the AMS, 43 (3) (2006), 305-364.

****Maple 11, User Manual*, Maplesoft, Waterloo Maple Inc., Waterloo, Ontario, Canada, 2007.

way to store the in-groupoid associated to a labeled directed graph is obtained by choosing a representative $w_{[u]}$ in each orbit [u] and storing all input isomorphisms from $w_{[u]}$ to $w_{[u]}$. For any other $v \in [u]$ it is enough to store one input isomorphism from v to $w_{[u]}$. A Maple procedure to construct this kind of structure associated to a labeled directed graph can be easily written using the preceding procedures. Also having such kind of structure, we can provide Maple procedures for computing all elements of the in-groupoid, the orbits, etc. We shall develop a complete package in joint wok with System Engineering students

REFERENCES

[1] M. Golubitsky, I. Stewart and M. Pivato, *Symmetry Groupoids and Patterns of Synchrony in Coupled Cell Networks*, Siam J. Applied Dynamical Systems, 2 (4) (2003), 609-646.

[2] M. Golubitsky, I. Stewart and A. Torok, *Patterns of Synchrony in Coupled Cell Networks with Multiple Arrows*, Siam J. Applied Dynamical Systems, 4 (1) (2005), 78-100.

[3] M. Golubitsky and I. Stewart, *Nonlinear dynamics of networks: the groupoid formalism*, Bulletin of the AMS, 43 (3) (2006), 305-364.

****Maple 11, User Manual*, Maplesoft, Waterloo Maple Inc., Waterloo, Ontario, Canada, 2007.