USING MAPLE TO REPRESENT THE SUBGROUPOIDS OF TRIVIAL GROUPOID X×Z×X

Mădălina Roxana **BUNECI**, University Constantin Brâncuşi of Târgu-Jiu, ROMÂNIA

Abstract. *The purpose of this paper is to present a way for studying subgroupoids G of X×**Z**×X (and especially, restrictions of the groupoids associated to discrete dynamical system) taking* advantage of *Maple symbolic and arbitrary precision computing capabilities.*

Keywords: trivial groupoid; discrete dynamical system; groupoid reduction; orbit; equivalence relation.
.

1. INTRODUCTION

In [3] we established necessary and sufficient conditions for a subgroupoid G of the trivial groupoid X×**Z**×X to be associated with a discrete dynamical system (X,φ). For the case of a finite set X we provided Maple procedures for testing if the subgroupoid G of X×**Z**×X arises from a discrete dynamical system and if the answer is positive to find all the time evolutions φ with the property that G(X,φ)⊂G. The approach used in this paper to represent the data of a subgroupoid G of X×**Z**×X allows us to treat the case of a general set X. More precisely, we prove that a subgroupoid G of X×**Z**×X is characterized by the set X and two functions f : X→X and k :X→**Z**. We shall use the notation and terminology from [1] and [3].

2. SUBGROUPOIDS OF X×Z×X DEFINED IN TERMS OF FUNCTIONS

In [3] we proved that any subgroupoid G of X×**Z**×X with $G^{(0)}$ = X can be obtained using an equivalence relation R on X and a family $\{k_{u,v}\}_{(u,v)\in R}$ of integer numbers satisfying the following conditions

1. $k_{u,u}\geq 0$ for all u∈X.
2. For all equivalent elements u,v,w∈X, if $k_{u,u} \neq 0$, then $k_{u,v} + k_{v,u} = k_{u,w} \pmod{k_{u,u}}$, else $k_{u,v} + k_{v,w} = k_{u,w}$.

R be the principal groupoid associated to G and

$$G =\{(u,k_{u,v}+tk_{u,u},v): (u,v)\in R,\ t\in \mathbf{Z}\}$$

For each orbit [u] = {v: (u,v)∈R} of R let us choose an element $f_{[u]}$ ∈[u]. Then we can define the following functions

$$f: X\to X,\ f(u)=f_{[u]} \text{ for all } u\in X$$

$$k: X\to \mathbf{Z},\ k(u)=k_{u,f(u)} \text{ for all } u\in X$$

The functions f and k defined above have the following properties:

1. f(f(u))=f(u) for all u∈X.
2. k(f(u)) ≥0 for all u∈X.

3. If $k(f(u)) \neq 0$, then $k(u) \in \{0,1,\ldots, k(f(u))-1\}$.

Conversely, any functions f and k with the above properties define a subgroupoid G of $X\times\mathbf{Z}\times X$ with $G^{(0)} = X$. Indeed,

$R=\{(u,v): f(u)=f(v)\}$ is an equivalence relation on X.

Let us define

$k_{u,u} := k(f(u))$ for all $u\in X$.

$$k_{u,v} := \begin{cases} (k(u)+k(f(u)-k(v)) \bmod k(f(u)), & \text{if } k(f(u))\neq 0 \\ k(u) - k(v), & \text{if } k(f(u)) = 0 \end{cases}$$

for all $(u,v)\in X\times X$ with the property that $f(u)=f(v)$ and $u\neq v$.

It is easy to see that R and $\{k_{u,v}\}_{(u,v)\in R}$ satisfy the conditions at the beginning of this section, and consequently, define a subgroupoid G

$$G = \{(u,k_{u,v}+tk_{u,u},v): (u,v)\in R,\ t\in\mathbf{Z}\}$$

of $X\times\mathbf{Z}\times X$ with $G^{(0)} = X$.

Example: Let $\varphi:X\to X$ be a function and

$$G(X, \varphi) = \{(u,k,v)\in X\times\mathbf{Z}\times X: \text{there is } n\in\mathbf{Z} \text{ such that } n\geq 0,\ n+k\geq 0 \text{ and } \varphi^{n+k}(u) = \varphi^{n}(v)\}.$$

Then $G(X, \varphi)$ is a subgroupoid of $X\times\mathbf{Z}\times X$ having the same unit space [4] ($G(X, \varphi)$ is the groupoid associated with a discrete dynamical system (X,φ)).

The procedures `cycle_detection and  equiv_detection` are modified versions of those defined in [1] (since X is not necessarily finite) and are based on Brent algorithm [2] to solve the cycle detection problem for given φ. The procedure `cycle_detection returns [0,0] if the isotropy group of` $G(X, \varphi)$ at u is $\{(u,0,u)\}$. Otherwise the procedure returns `[`k_u`,`n_u`] where` k_u `is` the smallest positive number such that there is $n\in\mathbf{N}$ with the property that $\varphi^{k_u+n}(u) = \varphi^{n}(u)$ and n_u is the smallest value $n_u\in\mathbf{N}$ with the property that $\varphi^{k_u+n_u}(u)=\varphi^{n_u}(u)$. The parameter nmax is used to avoid infinite iterations in the case of singleton `isotropy group` at u (nmax is maximum value of k_u we are looking for).

```
> cycle_detection:=proc(phi,u,nmax)
  local nu, ku, power, i, tortoise, hare;
     power:=1; ku:=1;
     tortoise:=u;
     hare:= phi(u);
     while(tortoise-hare<>0)  do
       if(power=ku) then
             tortoise:=hare;
             power:=power*2;
             ku:=0;
```

```
         end if;
        hare:=phi(hare);
        ku:=ku+1;
        if(ku>nmax)then RETURN([0,0]) end if;
        end do;
        nu:=0;
        hare:=u; tortoise:=u;
        for i from 0 to ku-1 do hare:=phi(hare) end do;
        while(tortoise-hare<>0)do
              tortoise:=phi(tortoise);
              hare:=phi(hare);
              nu:=nu+1;
        end do;
        RETURN([ku,nu])
  end proc;
  > cycle_detection(x->3.1*x*(1-x),0.5,1024);
                                     [2, 65]
  > ((x->3.1*x*(1-x))@@67)(0.5)=((x->3.1*x*(1-x))@@65)(0.5);
                          0.7645665203 = 0.7645665203
```

Let us make some remarks. If the isotropy group at u is {(u,kt,u):t∈**Z**} with k>nmax then the result of the above procedure is inexact.

If the dynamical system (X,φ) exhibits chaotic behavior, the floating-point arithmetic causes rounding errors which are magnified after each iteration step (hare:=phi(hare)). Symbolic computations allow you to obtain highly accurate results (in order to use symbolic computation the previous command should be written cycle_detection(x->31*x*(1-x)/10,1/2,1024)). Also in Maple approximations can be computed to any precision that is required (by setting the global variable Digits). However the symbolic computations and the theoretically "infinite precision" may need more time and space for a response.

It is not difficult to modify the procedure cycle_detect to compute the iterations of phi using the algorithm introduced in [5] (an arbitrary-precision floating-point approach based on automatic error analysis).

The procedure equiv_detection is similar to that define in [1] and it is applicable in the case of non-singleton isotropy groups. If the isotropy group of G(X, φ) at u is {(u,0,u)} and v∈[u], then procedure equiv_detection0 returns [$k_{u,v}$, $n_{u,v}$], where $k_{u,v}$ is the unique integer number with the property that there is n∈**N** such that $\varphi^{k_{u,v}+n}(u) = \varphi^{n}(v)$, and $n_{u,v}$ is the smallest natural value satisfying $\varphi^{k_{u,v}+n_{u,v}}(u) = \varphi^{n_{u,v}}(v)$.

```
> equiv_detection0:=proc(phi,u,v,nmax)
  local nuv,kuv,power,i,tortoise, hare,t_hare,test,p;
      test:=0;power:=1;kuv:=0;
      tortoise:=v; t_hare:=v;hare:= u; p:=0;
      while(tortoise<>hare) do
            hare:=phi(hare); t_hare:=phi(t_hare);
            p:=p+1;kuv:=kuv+1;
            if(power=kuv) then
                tortoise:=t_hare;
                power:=power*2;
                kuv:=0;
            end if;
            if(p>nmax) then test:=1; tortoise:=hare end if;
      end do;
      if(test=1) then
            power:=1;kuv:=0;
            p:=0;tortoise:=u; t_hare:=u; hare:= v;
            while(tortoise<>hare) do
                  hare:=phi(hare); t_hare:=phi(t_hare);
                  p:=p+1;kuv:=kuv+1;
                  if(power=kuv) then
                      tortoise:=t_hare; power:=power*2; kuv:=0;
                  end if;
                  if(p>nmax) then RETURN(NULL) end if;
            end do;
            hare:=v; tortoise:=u; else hare:=u; tortoise:=v;
      end if;
      nuv := 0;
      for i from 0 to kuv-1 do hare:=phi(hare) end do;
      while(tortoise<>hare)do
            tortoise:=phi(tortoise);
            hare:=phi(hare);
            nuv:=nuv+1;
      end do;
      if (test=0) then RETURN([kuv,nuv]) else RETURN([-kuv,nuv+kuv]) end if
end proc;

> equiv_detection0(x->4*x*(1-x),0.81,0.82,1024);

> equiv_detection0(x->4*x*(1-x),0.81,((x->4*x*(1-x))@@3)(0.81),1024);
                                [3, 0]
> equiv_detection0(x->4*x*(1-x),((x->4*x*(1-x))@@3)(0.81),0.81,1024);
                                [-3, 3]
```

For implementation in Maple of the reduction to $A=\{x_1,x_2,\ldots,x_n\}\subset X$ of a subgroupoid $G\subset X\times\mathbf{Z}\times X$ characterized by functions f (satisfying $f(A)\subset A$) and k we use a list L of three arrays:

L[1] contains a sequence obtained by sorting A and eliminating the duplicates,

L[2][i] = the index in L[1] of f(L[1][i]), i=1..n,

L[3][i] = k(L[1][i]), i=1..n.

The Maple procedure groupoid_data(phi) constructs the list L for the reduction to $A=\{x_1,x_2,\ldots,x_n\}\subset X$ of the groupoid $G(X,\varphi)$ associated with a function $\varphi:X\to X$.

```
> groupoid_data:=proc(phi,A,nmax)
 local i,j,fj,cd,n,L1,L2,Lij,fin,x,x1,ax,test,_n,m;
      n:=op(2,op(2,A)); x:=array(1..n);
      for i from 1 to n  do x[i]:=A[i] end do;
      fin:=n-1; m:=1;
      while (m<>0) do
            m:=0;
            for i from 1 to fin do
```

```
                if x[i]>x[i+1] then
                      ax:=x[i]; x[i]:=x[i+1]; x[i+1]:=ax; m :=i
                end if
          end do;
          fin:=m;
    end do;
    j:=1;
    for i from 1 to n-1 do
          if x[i]<x[i+1] then j:=j+1 end if
    end do;
    x1:=array(1..j); x1[1]:=x[1];j:=2;
    for i from 1 to n-1 do
          if x[i]<x[i+1] then x1[j]:=x[i+1]; j:=j+1 end if
    end do;
    n:=j-1;
    _n:=array(1..n);
    L1:=array(1..n); L2:=array(1..n);
    L1[1]:=1; cd:=cycle_detection(phi,x1[1],nmax);
    L2[1]:=cd[1];_n[1]:=cd[2];
    for i from 2 to n do
          cd:=cycle_detection(phi,x1[i],nmax);
          _n[i]:=cd[2];
          test:=0;
          j:=1;
          while j<i do
                fj:=L1[j];
                if(L2[fj]=cd[1]) then
                      if (cd[1]<>0) then
         Lij:=equiv_detection(phi,x1[i],x1[fj],2*cd[1]+max(cd[2],_n[fj]));
                      else
                         Lij:=equiv_detection0(phi,x1[i],x1[fj],2*nmax);
                      end if;
                      if(Lij<>NULL) then
                          L1[i]:=L1[fj]; L2[i]:=Lij[1];
                          test:=1; j:=i
                      end if
                end if;
                j:=j+1;
          end do;
          if test=0 then L1[i]:=i; L2[i]:=cd[1] end if
    end do;
    RETURN([evalm(x1),evalm(L1),evalm(L2)])
end proc;
> Lphi1:=groupoid_data(x->4*x*(1-x),array([seq(i/100.,i=0..100)]),100):
> seq([i,Lphi1[2][i],Lphi1[3][i]],i=1..op(2,op(2,Lphi1[1])));
[1, 1, 1], [2, 2, 0], [3, 3, 0], [4, 4, 0], [5, 5, 0], [6, 6, 0], [7, 7,
0], [8, 8, 0], [9, 9, 0], [10, 10, 0], [11, 3, 1],[12, 12, 0], [13, 13, 0],
[14, 14, 0], [15, 15, 0],[16, 16, 0], [17, 17, 0], [18, 18, 0], [19, 19,
0],[20, 6, -1], [21, 3, 1], [22, 22, 0], [23, 23, 0],[24, 24, 0], [25, 25,
0], [26, 26, 1], [27, 27, 0],[28, 28, 0], [29, 29, 0], [30, 30, 0], [31,
17, 1],[32, 32, 0], [33, 33, 0], [34, 34, 0], [35, 35, 0],[36, 10, 1], [37,
3, 0], [38, 38, 0], [39, 39, 0],[40, 40, 0], [41, 5, 1], [42, 42, 0], [43,
43, 0],44, 44, 0], [45, 45, 0], [46, 2, 1], [47, 47, 0],[48, 48, 0], [49,
49, 0], [50, 16, -1], [51, 1, 1],[52, 16, -1], [53, 49, 0], [54, 48, 0],
[55, 47, 0],[56, 2, 1], [57, 45, 0], [58, 44, 0], [59, 43, 0],[60, 42, 0],
[61, 5, 1], [62, 40, 0], [63, 39, 0],[64, 38, 0], [65, 3, 0], [66, 10, 1],
[67, 35, 0],[68, 34, 0], [69, 33, 0], [70, 32, 0], [71, 17, 1],[72, 30, 0],
[73, 29, 0], [74, 28, 0], [75, 27, 0],[76, 26, 0], [77, 25, 0], [78, 24,
0], [79, 23, 0],[80, 22, 0], [81, 3, 1], [82, 6, -1], [83, 19, 0],[84, 18,
0], [85, 17, 0], [86, 16, 0], [87, 15, 0],[88, 14, 0], [89, 13, 0], [90,
12, 0], [91, 3, 1],[92, 10, 0], [93, 9, 0], [94, 8, 0], [95, 7, 0], [96, 6,
0],[97, 5, 0], [98, 4, 0], [99, 3, 0], [100, 2, 0], [101, 1, 0]
```

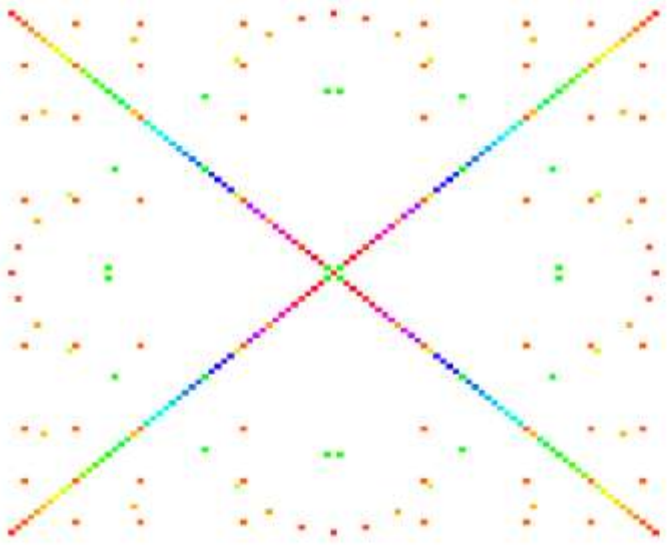
3. USING MAPLE TO STUDY SUBGROUPOIDS OF X×Z×X

In this section we provide some example of Maple procedure for studying a subgroupoid G of X×**Z**×X.

The procedure orbits displays the graph of the equivalence relation (principal groupoid) associated with the groupoid G characterized by gd (list of three arrays).

```
> orbits:=proc(gd)
  local i,j,n,m,no,elem;
      n:=op(2,op(2,gd[1]));
      no:=gd[2][1];
      for i from 2 to  n do
            if gd[2][i]>no then no:=gd[2][i] end if
      end do;
      elem:=array(1..n*n);
      m:=0;
      for i from 1 to n do
            m:=m+1;
                                          elem[m]:=rectangle([i-1,i],[i,i-
                                 1],color=COLOR(HSV,gd[2][i]/no,1.,1.));
            for j from i+1 to n do
                  if gd[2][i]=gd[2][j] then
                        m:=m+1;
                                          elem[m]:=rectangle([i-1,j],[i,j-
                                 1],color=COLOR(HSV,gd[2][i]/no,1.,1.));
                        m:=m+1;
                                          elem[m]:=rectangle([j-1,i],[j,i-
                                 1],color=COLOR(HSV,gd[2][i]/no,1.,1.));
                  end if
            end do
      end do;
      RETURN(display(seq(elem[i],i=1..m),axes=none,style=patchnogrid))
end proc;

> orbits(groupoid_data(x->4*x*(1-x),array([seq(i/100.,i=0..100)]),100));
```



The procedure `compute_kuv` returns $k_{u,v}$ (the smalest integer number with the property that there is n∈**N** such that $\varphi^{k_{u,v}+n}(u) = \varphi^{n}(v)$) using

$$k_{u,u} := k(f(u)) \text{ for } u = x_i = x_j$$

$$k_{u,v} := \begin{cases} (k(u)+k(f(u)-k(v)) \bmod k(f(u)), & \text{if } k(f(u)) \neq 0 \\ k(u) - k(v), & \text{if } k(f(u)) = 0 \end{cases}$$

for u = x_i and v=x_j (i≠j)

```
> compute_kuv:=proc(gd,i,j)
  local kij;
      if gd[2][i]<>gd[2][j] then RETURN(NULL) end if;
      if i=j then
            kij:=gd[3][gd[2][i]]
      else
            if gd[3][gd[2][i]]<>0 then
                  kij:=irem(gd[3][i]+gd[3][gd[2][i]]-
            gd[3][j],gd[3][gd[2][i]]) else
                  kij:=gd[3][i]-gd[3][j]
            end if;
      end if
end proc;
> compute_kuv(groupoid_data(x->4*x*(1-
x),array([seq(i/100.,i=0..100)]),100), 2, 100);
                                                0
```

The procedure `iso_k` returns an one-dimensional array containing the indices of the units u∈A={x_1,x_2,…,x_n}⊂X with the property that the isotropy group at u is {(u,kt,u),t∈**Z**}.

```
> iso_k:=proc(gd,k)
 local a1,a2,i,n,j;
      n:=op(2,op(2,gd[1]));
      a1:=array(1..n);
      j:=0;
      for i from 1 to n do
            if gd[3][gd[2][i]]=k then j:=j+1;a1[j]:=i end if
      end do;
      a2:=array(1..j);
      for i from 1 to j do a2[i]:=a1[i] end do;
      RETURN(evalm(a2))
end proc;
> iso_k(groupoid_data(x->4*x*(1-x),array([seq(i/100.,i=0..100)]),100,1);
                         [1, 26, 51, 76, 101]
```

Thus for u∈{0, 0.25, 0.5, 0.75, 1} the isotropy group at u isomorphic to **Z**.

Assuming that the isotropy groups at u∈A of the groupoid G(X,φ)|A are not singleton, the procedure `saturation` constructs a set S satisfying the properties: A⊂S and φ(S)⊂S.

```
saturation:=proc(phi,A,nmax)
local n,m,i,j,_k,_n,cd,sat;
      n:=op(2,op(2,A));_k:=array(1..n); _n:=array(1..n);m:=0;
      for i from 1 to n do
            cd:=cycle_detection(phi,A[i],nmax);
            _k[i]:=cd[1];_n[i]:=cd[2];
            if (_k[i]<>0) then m:=m+_k[i]+_n[i];
            else m:=m+nmax+1
            end if;
      end do;
      sat:=array(1..m); m:=0;
      for i from 1 to n do
            m:=m+1;sat[m]:=A[i];
            if _k[i]<>0 then
                  for j from 1 to _k[i]+_n[i]-1 do
                        m:=m+1; sat[m]:=phi(sat[m-1])
                  end do;
            else
                  for j from 1 to nmax do
                        m:=m+1; sat[m]:=phi(sat[m-1])
                  end do;
            end if
      end do;
      RETURN(evalm(sat))
end proc;
```

However the dimension of the array returned by the procedure `saturation` could be appreciably larger then the dimension of A.

```
> iso_k(groupoid_data(x->3.2*(1-
x)*x,array(1..9,[seq(i/8.,i=0..8)]),100),0);
```

```
                                       [ ]
> op(2,op(2,saturation(x->3.2*x*(1-
x),array(1..9,[seq(i/8.,i=0..8)]),100)));
                                       212
```

Furthermore all procedures provided in [3] can be modified according the representation of the groupoid data used in this paper.

REFERENCES

[1] I. C. Bărbăcioru and M. Buneci, *Using Maple to study the groupoid associated to a singly generated dynamical system. I,* Annals of the "Constantin Brâncuşi" University of Târgu-Jiu. Egineering Series. No. **3**(2010), 308-317.

[2] R. P. Brent, *An improved Monte Carlo factorization algorithm,* BIT **20** (1980) 176–184.

[3] M. Buneci, *Groupoid reductions associated to discrete dynamical systems,* Annals of the "Constantin Brâncuşi" University of Târgu-Jiu. Engineering Series. No. **3**(2012), 171-182.

[4] J. Renault, *Cuntz-like algebras*, in Operator theoretical methods (Timişoara, 1998), 371-386, Theta Found., Bucharest, 2000.

[5] C. Spandl, *Computational complexity of iterated maps on the interval*, Mathematics and Computers in Simulation, **82** (8) (2012) 1459–1477.