SCIARA – Formal Definition

SCIARA: Simulation by **C**ellular **I**nteractive **A**utomata of the **R**heology of **A**etnean lava flows

$$SCIARA = <R, X, L, Q, P, \sigma, \gamma>$$

R = $\{(x\ y) | x, y \in N, 0 \le x \le lx, 0 \le y \le ly\}$ is the set of points with integer co-ordinates in the finite region, where the phenomenon evolves. *N* is the set of natural numbers.

L$\subseteq$**R** specifies the lava source cells

• **X** = {(0,0), (0,1), (0,-1), (1,0), (-1,0)} is the set, which identifies the geometrical pattern of the cells, which influence the cell state change.

• The finite set **Q** of states of the ea: $\mathbf{Q = Q_a \times Q_{th} \times Q_T \times Q_o^4 \times Q_i^4}$

Q_a	altitude of the cell
Q_{th}	lava thickness in the cell
Q_T	lava temperature in the cell
$Q_o(Q_j)$	lava outflow (inflow)

SCIARA Parameters

P is the set of *global parameters* of SCIARA

$$P = \{p_l, p_t, p_{adh\_v}, p_{adh\_i}, p_{adh\_s}, p_{Tv}, p_{Ti}, p_{Ts}, p_r, p_c\}$$

Optimized by GAs

p_l	side of the cell	5 m
p_t	temporal correspondence of a step of SCIARA	60 s
p_{adh_v}	lava adhesion at the vents	0.7 m
p_{adh_i}	lava intermediate adhesion	1.0 m
p_{adh_s}	lava adhesion at the solidification	10 m
p_{Tv}	lava temperature at the vents	1100 °C
p_{Ti}	Lava intermediate temperature	1050 °C
p_{Ts}	lava temperature at solidification	850 °C
p_r	relaxation rate	1
p_c	cooling parameter	$1.4\ 10^{-14}$ $(m/K)^3$

Substates for SCIARA

The finite states for EACH cell are:

1. **Cell Altitude** (e.g. 2000m): varies according to lava solidification;
2. **Lava Width** (e.g. 6 m): varies according to incoming and outcoming flows
3. **Lava Temperature** (e.g. 1200 K°): varies according to average temperature and energy losses at surface
4. **Lava Flows** (e.g. 4), towards neighbouring cells: calculated using the "minimisation algorithm"

SCIARA (release hex1) transition function σ

1) Internal transformation: SOLIDIFICATION

$$\sigma_S: Q_{th} \times Q_a \times Q_T \rightarrow Q_a \times Q_{th}$$

The cell altitude remains unchanged until solidification condition holds:

$$(Q_T < p_{Ts})$$

then the altitude is increased by the lava thickness and lava thickness is zeroed.

SCIARA (release hex1) transition function σ

2) Local interaction: LAVA OUTFLOWS

$$\sigma_{LO}: (Q_{th} \times Q_a)^5 \times Q_T \rightarrow Q_o^4$$

Adhesion computation: $adhesion = f(p_{adh\_v}, p_{adh\_s}, p_{Tv}, p_{Ts}, Q_{th}, Q_T)$

Minimisation algorithm with:

q_d = quantity, that may be distributed, in the central cell = $Q_{th} - adhesion$

q_0 = irremovable quantity in the central cell = $Q_a + adhesion$

q_i = quantity in the cell i = $Q_a + Q_{th}$ **1≤ i ≤ 4**

SCIARA (release hex1) transition function σ

3) Local interaction: LAVA MIXING $\sigma_{LM}: Q_{th} \times Q_i^4 \times Q_o^4 \times Q_r^5 \rightarrow Q_r$

Lava mixing involves the determination for the central cell:

a) the remaining lava thickness (*rem_th*):

$$rem\_th = Q_{th}[0] - \Sigma_j Q_o[j] \qquad 1 \le j \le 4$$

b) new lava thickness (*new_th*):

$$new\_th = rem\_th + \Sigma_j Q_i[j] \qquad 1 \le j \le 4$$

c) the temperature variation by mixing is calculated as the average weight of Q_T, by considering both the remaining lava and the inflows:

$$new\_T = (rem\_th * Q_T[0]) + \Sigma_j (Q_i[j] * Q_T[j])) / (rem\_th + \Sigma_j Q_i[j])$$

$$1 \le j \le 4$$

SCIARA (release hex1) transition function σ

4) Internal transformation: LAVA COOLING $\sigma_{LC}: Q_{th} \times Q_T \rightarrow Q_T$

Temperature drop due to irradiation at the surface is computed, assuming that other losses are not relevant

$$new\_T = Q_T \Big/ \sqrt[3]{1+(Q_T^3 \cdot p_c)}$$

SCIARA Camelot CA Definition

```
cadef
{
   dimension 2;
   region All_AC(:,:,:);
   radius 1;
   state (float quote;
          float width,
                max_width,   // needed for Genetic Algorithm
                temperature,
                flow[4];
          float vent_rate,
                real,
                RandS,
                RorS);

   neighbor news[4]([0,-1]NORTH,[1,0]EAST, [-1,0]WEST,[0,1]SOUTH);

   parameter(prm_lato      10.0,
             prm_clock     60.0,
             prm_admin      0.0,
             prm_admid      0.0,
             prm_admax      0.0,
             prm_tcrat      0.0,
             prm_tmid       0.0,
             prm_tsolid     0.0,
             prm_cool       0.0,
             prm_rall       0.0,
             prm_days       1.0,
             prm_maxstep    5760.0
   );
}//cadef
```

sciara.cpt (Main)

```
// main
{
  switch (step%2) {
        case 0:
            calc_flows();
        break;

        case 1:
          calc_width();
          calc_temperature();
          calc_quote();
          if (step < 2 * prm_clock * 24 * prm_days)
              vent();
        break;

  }//switch
```

Fitness Function

```
if (step == (int) prm_maxstep - 1)
 {
  if (cell_max_width > 0.0 && cell_real > 0.0)

       update(cell_RandS, 1.0);

  if (cell_max_width > 0.0 || cell_real > 0.0)

       update(cell_RorS, 1.0);
```

New lava width

```
/* Calculate new lava width*/

void calc_width()
{
  int   i;
  float new_width;

  new_width=cell_width;
  for (i=0; i<4; i++)
    new_width-=(cell_flow[i]-news[i]_flow[3-i]);
  update(cell_width,new_width);
  if (new_width > cell_max_width)
     update(cell_max_width, new_width);


} // calc_width
```

Elementary processes:Lava Cooling and Solidification

A **two step** process determines the new cell temperature. In the first one, the cell temperature is obtained as **weighted average** of residual lava inside the cell and lava inflows from neighbouring ones:

$$T_{av} = \left(t_r \times T(0) + \sum_{i=1}^{6} f(i,0) \times T(i)\right) \Big/ \left(t_r + \sum_{i=1}^{6} f(i,0)\right)$$

where $t_r \in Q_t$ is the residual lava thickness inside the central cell after the outflows distribution, $T \in Q_T$ is the lava temperature and $f(i,0)$ the lava inflow from the i^{th} neighbouring cell.

Note that $f(i,0)$ is equal to the lava outflow from the i^{th} neighbouring cell towards the central one, computed by means of the **minimisation algorithm**

Elementary processes:Lava Cooling and Solidification

The final step updates the previous calculated temperature by considering **thermal energy loss** due to lava surface irradiation:

$$T = T_{av} \Big/ \sqrt[3]{1 + \left(T_{av}^3 CA \Big/ V\right)}$$

where C is a parameter depending on lava rheology, A is the surface area of the cell, and V the lava volume

Elementary processes:Lava Cooling and Solidification

```
void calc_temperature()
{
  int   i;
  float sommah,
        sommath,
        new_temp;

    sommah=cell_width;
    for (i=0; i<4; i++)
      sommah-=cell_flow[i];

    //weighted average
    sommath=sommah*cell_temperature;
    for (i=0; i<4; i++){
      sommah+=news[i]_flow[3-i];
      sommath+=(news[i]_temperature*news[i]_flow[3-i]);
    }//for
    //cooling
    if (sommah>0.){
      new_temp=sommath/sommah;
      new_temp/=pow(1.+pow(new_temp,3.)*prm_cool,1./3.);
      update(cell_temperature,new_temp);

    }//if
    //else
     //update(cell_temperature,prm_tsolid);

} // calc_temperature
```

Elementary processes:Lava Cooling and Solidification

Lava Solidification. When the lava temperature drops below the threshold T_{sol}, lava solidifies. Consequently, cell altitude increases by an amount equal to lava thickness and new lava thickness is set to zero.

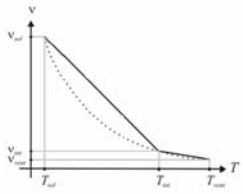
```
/* Calculate new lava quote*/

void calc_quote()

{
  if ((cell_temperature<=prm_tsol)&&(cell_width>0)){
      //solidification...
    update(cell_quote,cell_quote+cell_width);
    update(cell_width,0);
    update(cell_temperature,prm_tsol);
 }//if

}//calc_quote
```

Lava Adherence

Lava Flows. Lava rheological resistance **increases** as temperature **decreases**; consequently, a certain amount of lava, i.e. the lava adherence v, cannot flow out from the central cell towards any neighbouring ones. It is obtained by means of a **piecewise linear function** that approximates the equation

$$v = k_1 e^{-k_2 T}$$

where $T \in Q_T$ is the cell lava temperature, while k_1 and k_2 are parameters depending on lava rheological properties



Lava Adherence

```
/* Calculate lava adherence*/

float calc_adherence()

{
  float coeff,
        ad;

  //first part of the linear function
  if ((cell_temperature<=prm_tcrat)&&(cell_temperature>=prm_tmid)){
    coeff=(prm_admid-prm_admin)/(prm_tmid-prm_tcrat);
    ad=coeff*(cell_temperature-prm_tcrat)+prm_admin;
  }//if

   else{  //second part of the linear function
    coeff=(prm_admax-prm_admid)/(prm_tsolid-prm_tmid);
    ad=coeff*(cell_temperature-prm_tmid)+prm_admid;
   }//else

  return ad;

} //calc_adherence
```

Outflow calculation

The Minimisation Algorithm of the Differences.

The algorithm computes the outflows in order to minimise the differences of "**certain quantities**" in the neighbourhood. It is based on the following assumptions:

- two parts of the considered quantity must be identified in the central cell: these are the unmovable part, $u(0)$, and the mobile part, m;
- only m can be distributed to the adjacent cells. Let $f(a,b)$ denote the flow from cell a to cell b; m can be written as:

$$m = \sum_{i=0}^{\#X} f(0,i)$$

where $f(0,0)$ is the part which is not distributed;

Outflow calculation

- the quantities in the adjacent cells, $u(i)$ (i=1,2,...,#X) are considered **unmovable**;
- let $c(i)=u(i)+f(0,i)$ (i=0,1,...,#X) be the new quantity content in the i^{th} neighbouring cell after the distribution; let c_{min} be the minimum value of $c(i)$ (i=0,1,...,#X).

The outflows are computed in order to **minimise** the following expression:

$$\sum_{i=0}^{\#X}(c(i)-c_{\min})$$

This represents the condition of **maximum possible equilibrium** for the considered quantity in the neighbourhood, according to the **principle of hydrostatic equilibrium.**

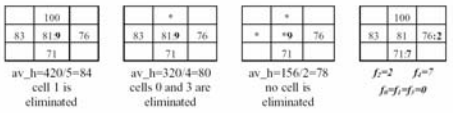
Moreover, a **relaxation rate** $r_r \in [0,1]$ can be introduced, denoting that such conditions may not be reached in a single CA step; the obtained values of outflows are therefore multiplied by r_r (if r_r=1, no relaxation is induced; if r_r=0, there will be no outflows towards the neighbourhood).

Eventually, the **simultaneous application** of the minimization principle to each cell gives rise to the **global equilibrium** of the system.

Outflow calculation

EXAMPLE OF DISTRIBUTION (von Neumann neighbouring):

q_4 = 9, q_0 = 81, q_1 = 100, q_2 = 76, q_3 = 83, q_4 = 71,



Outflow calculation

```
/* Calculate outgoing flows*/

void calc_flows()

{
  char wlog[5]={TRUE,TRUE,TRUE,TRUE,TRUE},
       elim;
  int i,
      k,
      kk;
  float adherence,
        wqc[5],
        distrc,
        qav,
        rall;

   adherence=calc_adherence();
   if ((adherence<cell_width)&&(cell_width>0))
    {
      wqc[0]=cell_quote+adherence;
      for (i=1; i<5; i++)
        wqc[i]=news[i-1]_width+news[i-1]_quote;
      distrc=cell_width-adherence;

// continue…
```

Outflow calculation

```
// calculate outgoing flows (minimization algorithm)
     do{
       elim=FALSE;
       qav=distrc;
       kk=0;
       for (k=0; k<5; k++)
         if (wlog[k]){
           qav+=wqc[k];
           kk++;
         }//if
       if (kk!=0)
         qav/=kk;
       for (k=0; k<5; k++)
         if((qav<=wqc[k])&&(wlog[k])){
           wlog[k]=FALSE;
           elim=TRUE;
         }//if
     }while (elim);
     for (k=1; k<5; k++)
       if (wlog[k]){
         update(cell_flow[k-1],(qav-wqc[k])*prm_rall);
       }//if
       else
         update(cell_flow[k-1],0.);
   }//if
   else
     for (k=1; k<5; k++)
      update(cell_flow[k-1],0.);
} //calc_flows
```

Steering

- The steering function is an optional feature of a CARPET program by which the user can affect the flow of the program as a result of global reductions on regions of the model
- The steering function is defined in a separate section of the CARPET program, similarly to the update function. *The main difference is that the update function is applied separately in each cell, whereas the steering function is global for the model.* Any code inside the steering statement is copied verbatim to the generated file, with the exception of the `region_<op>()` statements which are translated to a global reduction function.

Steering - 1

```
steering
{
  if (step == 0)
  {
    randS = 0.0; rorS = 0.0;
    apotema=0.866025*prm_lato;

    f = fopen("param.txt", "r");
    fscanf(f, "%s", s); fscanf(f, "%s", s);
    cpt_set_param(&prm_admin,   atof(s));
    fscanf(f, "%s", s); fscanf(f, "%s", s);
    cpt_set_param(&prm_admid,   atof(s));
    fscanf(f, "%s", s); fscanf(f, "%s", s);
    cpt_set_param(&prm_admax,   atof(s));
    fscanf(f, "%s", s); fscanf(f, "%s", s);
    cpt_set_param(&prm_tcrat,   atof(s));
    fscanf(f, "%s", s); fscanf(f, "%s", s);
    cpt_set_param(&prm_tmid,    atof(s));
    fscanf(f, "%s", s); fscanf(f, "%s", s);
    cpt_set_param(&prm_tsolid,  atof(s));
    fscanf(f, "%s", s); fscanf(f, "%s", s);
    cpt_set_param(&prm_cool,    atof(s));
    fscanf(f, "%s", s); fscanf(f, "%s", s);
    cpt_set_param(&prm_rall,    atof(s));

    fclose(f);

  }  // if…
```

Steering - 2

```
if (step == (int) prm_maxstep - 1)
 {

  randS = region_sum(All_AC, RandS);
  rorS = region_sum(All_AC, RorS);


  e1 = sqrt(randS / rorS);

  f = fopen("fitness.txt", "w");

  fprintf(f, "%f\n", e1);

  fclose(f);


 }
```

sciara_pga.c

- To launch the genetic algorithm

```
./nohup sciara_pga &
```