Advanced Cell-DEVS Models of Dendritic Growth in Supercooled Liquids

Daniella Niyonkuru[1] **Damián Vicino**[1, 2]
[1]Dept. of Systems and Computer Engineering
Carleton University 1125 Colonel By Dr., Ottawa, ON, Canada K1S 5B6
[2]INRIA Sophia Antipolis, Université Nice Laboratoire I3S UMR CNRS 7271, Sophia Antipolis, France
{Daniella.Niyonkuru, Damian.Vicino}@carleton.ca

ABSTRACT
2D Cell-DEVS models were developed to study both free and constrained dendritic growth in supercooled liquids. The defined models include rules that consider heat diffusion, the influence of the curvature on the equilibrium freezing temperature and latent heat evolution. Previously, we implemented a 3D supercooling model showing macroscopic structures and observed that the velocity of the dendrite tip growth was proportional to the supercooling temperature. In this paper, we focus on microstructures and analyze parameters that affect them. Our new simulations show primary and secondary dendritic arms, as well as the impact of different parameters.

1. INTRODUCTION

From causing airplanes accidents [1] [2] to creating perfect slush beverages and preserving organs [3], supercooling [4] can be both beneficial and harmful. Supercooling [5], also referred to as undercooling, is the process of lowering the temperature of a liquid below its freezing temperature without it becoming solid. In a pure liquid such as distilled water, particles (also called nucleation sites) responsible of solidification are missing. Usually, crystallization occurs around those small nuclei that are often impurity particles. The first crystal has the shape, for instance face-centered cubic for silver (Ag), into which the liquid will naturally solidify. Then, as the crystal grows, it tends to develop spikes and its shape changes into a tree-like form called a dendrite.

Dendrites [6] [7] are usually observed in materials solidifying with low entropies of fusion and freeze such that dendrite arms grow in specific crystallographic directions. As a dendrite grows, additional side arms can grow behind the growing tip. Moreover, dendrites can be either free or constrained. They are termed free dendrites when they form individually and grow in supercooled liquids. In this case, growth happens radially leading to an equiaxed shaped grain until collision with another growing dendrite takes place. Constrained dendrites, on the other hand, grow from a surface in a columnar fashion, such that the primary dendrite arms are in the direction of the heat flow.

Over the past fifty years, numerous attempts were made to study both analytically and experimentally the characteristics of dendritic growth since they play a major role in determining the final quality and properties of a solidified structure, especially metal alloys. However, the dendritic microstructures usually require complex computations and remain hard to study. In our previous work, we have shown the relationship between the undercooling temperature and the crystal growth speed, as well as macroscopic features. For this work, we will focus on the microscopic aspect that is the growth of the crystal and exhibit dendritic features. To achieve this goal, we have built several Cell-DEVS models from the most basics to complex ones using the extended CD++ tool [8] to render the behavior of dendritic growth and observe their microscopic characteristics. Our models implement the solidification process as well as the heat conduction that influence dendritic growth. We were able to observe both primary and secondary dendritic arms, and have run experiments for both free and constrained dendritic growth.

In this paper, we will first revisit the factors that influence dendritic growth and briefly review some existing models. Then, we will present the newly defined models, describe the experiments, and show results. Finally, we will conclude this paper with the analysis of the obtained results.

2. BACKGROUND

As introduced in the previous section, the evolution of a dendritic crystal depends on the complex interaction of several physical phenomena that includes latent heat evolution and its removal from the solid-liquid interface, solid-liquid interfacial energy, the influence of solid-liquid interface curvature on the equilibrium freezing temperature and the atomic mechanism of the crystal growth process. Two factors in particular control the growth of the dendrites: Heat liberated at the solid/liquid interface, and the local curvature of the solid/liquid interface. Therefore, both the heat transfer and the dendrite morphology problems are influenced by these conditions: the first is the local equilibrium freezing temperature, determined by the local interface curvature, and known as the Gibbs-Thomson effect. The second condition is that latent heat released when freezing must be removed from the solid liquid interface.

Most numerical methods that implemented the above factors were not able to simulate the oscillatory formation and growth of dendrite arm branches [9]. Afterwards, 2D

rule-based models (Cellular Automata - CA) were proposed to show not only the macroscopic patterns, but also simulate dendritic branches formation. The first elementary model uses window automata to simulate dendritic growth by applying simple rules where the number of solid neighbors should neither bee too high nor too low in order for a site to freeze [10]. This model, however, does not take into account the temperature factor. Packard later introduced a model that adds the temperature by integrating a continuous variable at each site to simulate heat transfer. As the degree of influence of curvature on the freezing temperature was reduced, the growth forms changed from amorphous to tendril like (tip splitting) to dendritic like structures exhibiting side branching [11]. For this paper, we will implement, using the extended Cell-DEVS syntax, the window automata model [10], add improvements made by Packard [11] and then proceed to the experiences presented in [9] that studies columnar and free dendritic growth.

2.1. Modelling Dendritic growth with CA – A review

2.1.1. Window Automata and Dendritic Growth

A crystal grown from a seed in solution can develop lacy dendritic shapes. This type of crystalline growth occurs when the seed is much colder than the surrounding solution. The heat dissipated by the crystallization process leads to the growth of dendrites that spread out into the solution to find colder zones of the liquid. This type of growth can be modelled by the use of "window" cellular automata [10]. The need to dissipate the heat generated by crystallization is handled by not allowing an automaton to change to state 1 if the number of its neighbors in state 1 is too large. The growth process of course requires the presence of a seed, and so a transition to state 1 cannot occur if the number of neighbor in state 1 is too small. The combination of these two tendencies means that a state 0 automaton can change its state to 1 only if the number of its neighbors in state 1 is neither too large or too small.

2.1.2. Packard Model

The Packard model [11] adds growth restriction rules related to the sum of neighboring sites. For dendritic growth, he added growth inhibition rules that depend on the local equilibrium temperature. The author defines a 2D Cellular Automaton with two states per cell and transition rules. The states denote presence or absence of solid, and the rules depend on their neighbors only through their sum. Four types of behavior can be observed:

- No growth
- Plate structure reflecting the lattice structure
- Dendritic structure with side branches growing along lattice directions
- Growth of an amorphous, asymptotically circular form

Two important ingredients are needed for correct dendritic behavior modelling:

- The Flow of Heat – modeled by addition of a continuous variable at each lattice site to represent temperature, and
- The Effect of Solidification on the Temperature Field – when solid is added to a growing seed, the latent heat of solidification must be radiated away.The temperature is set to a constant high value when new solid is added.

The model include hybrid of discrete and continuum elements. Different parameters can be used to study dendritic growth the :

- diffusion rate,
- latent heat added upon solidification, and the
- local temperature threshold

Packard shows few results to compare with when reproducing but the specific experiment parameters are not given. We will also show the impact of additional parameters.

2.1.3. Columnar and Free Dendritic Growth Simulation with Rule-Based Lattice Models (By Brown [9])

In [9], a cellular automaton model is presented and will be the reference for this study. An orthogonal grid will be used. Each cell will have its own temperature and state/phase. Initial temperatures will be set to supercooling values for each cell, and top and bottom borders will be insulated while side borders are set to be in contact with each other. It is assumed that dendrites were thermal in nature, and both constrained and free growths are studied.

2.1.5. Other Models

In [12], two Cellular Automata models are proposed based on the Moore and Von Neuman neighborhoods. In both cases, the neighbors are not only affected by the state of the near neighbors but also the extended neighbors, i.e. neighbors of each neighboring cell. This model defines three possible states: solid, characterized by a unit value; near-solid that are liquid cells with at least one solid neighbor; and liquid, liquid cells with no solid neighbors. Each cell value represents the concentration of solid particles. For instance, solid cells have a 100% concentration. Depending on the category, different equations are applied to obtain the next state of the cell. If the state is solid, it remains solid. In the case of a liquid cell, a diffusion formula is used to compute the new concentration. For near-solid cells, the same formula is used with an extra increment since solid particles are diffusing from the solid neighbors. The formula requires two parameters, alpha and beta, that respectively represent the weight diffusion and the constant increment (for near-solid cells). A particularity of this model is that the temperature is not set in a direct manner when simulating. The two types of neighborhood produce different results. In particular, the Moore (see Figure 19) model produce results with a 45 degrees rotation shift comparatively to the Von Neumman.

The paper includes several examples and parametrization values to reproduce the experiments.

In [13] a more complex approach is taken using partial solidification for each cell and several additional parameters. The model evaluates 21 equations per step for each cell. These equations include some partial differential equations. The approach uses Virtual Front Tracking, takes in account several microscopic phenomena such as the crystallographic orientation and alloys concentration.

[14] reviews previous models and suggests a new approach based on thermal fields. In this case, the model analyzes mesh-induced anisotropy for the traditional capture rules such as Von Neumann's and Moore's capture rules and simulate the tip growth velocity.

2.2. Cell-DEVS

An alternative formalism, Cell-DEVS [15], can be used to model this kind of behaviours. This formalism combines the advantages of Cellular Automata with those from Discrete-Event System Specification (DEVS). This approach provides more flexibility at the time of modelling cells and allows the combination of models defined using different formalisms.

In Cell-DEVS, each cell is defined as a Cell-DEVS atomic model, which is formally defined by the following tuple:

$$< X, Y, S, N, type, d, \tau, \delta_{int}, \delta_{ext}, \lambda, \text{ta} >$$

where:

- *X* is the set of input external events;
- *Y* is the set of output external events;
- *S* is the set of states;
- *N* is the set of input values
- *type* is the type of delay (transport, inertial, other);
- *d* is the delay for the cell;
- τ is the local computing function;
- δ_{int} is the internal transition function;
- δ_{ext} is the external transition function;
- λ is the output function; and
- *ta* is the time advance function.

To combine the cells a Cell-DEVS coupled model is necessary, this model is defined as the tuple

$$< X, Y, Xlist, Ylist, \eta, N, \{m, n\}, C, B, Z, select >$$

where:

- *X* is the set of input external events;
- *Y* is the set of output external events;
- *Xlist* are the list of input couplings;
- *Ylist* are the list of output couplings;
- η is the neighbourhood size;
- *N* is the neighbourhood set;
- *{m, n}* is the size of the cell space;
- *C* is the cell space set;
- *B* is the border cell set;
- *Z* is the translation function; and
- *select* is the tie breaking function

Each cell is an atomic DEVS model; it receives inputs from neighboring cells and sends output to its neighbors after defined time delays. After the behavior of a cell has been defined and proper delays specified, two essential aspects need to be delineated: the neighbors and the rules. The neighbors define the input sources of the cell and this later uses this compute its local computing function. The rule section defines the local computing function using this format:

$$\{Postcondition\}\ \ Delay\ \{Precondition\}$$

If state variables are used, the assignment portion can also be added to the previous format:

$$\{Postcon.\}\ \{Assignment\}\ Delay\ \{Precon.\}$$

In the following sections, the previously described dendritic models will be modeled using Cell-DEVS. Advanced Cell-DEVS [15], in particular, offers multiple state variables and neighboring ports that will be convenient for implementation. As mentioned previously, several factors are involved in order to simulate accuratelydendritic growth.

3. MODELS DEFINITION

3.1. Window Automata and Dendritic Growth

3.1.1. Cell-DEVS Formal Specification

The formal specification is described as follows:

```
CD = < X, Y, I, S, θ, N, d, δint, δext, τ, λ,
D >
X = Ø
Y = Ø
S = {0,1}
N = {(0,-1),(0,1),(0,0),(1,0),(-1,0)}
d = 100 ms
τ: N→S:
S = 1 if ((0,0)=0 and truecount = 1) or
((0,0)=1)
S = 0 in other cases
```

3.1.2. Neighborhood and Rules

Assuming the von Neumann neighborhood, the following transition rule will result in interesting dendritic growth patterns:

1. An automaton in state 1 will always stay in state 1.
2. An automaton in state 0 changes to state 1 only if exactly one of its neighbor is in state 1.

Therefore, if only one of the up, down, left or right neighbour is solid, a liquid cell will transition to solid. Once a cell is solid, it remains solid.

3.1.3. Cell-DEVS Implementation

This version uses the extended CD++ syntax, and has a state variables value for the cell state. A value of 1 means that the cell is solid, and a value of 0 indicated a liquid cell.

```
[top]
components : 2dsupercooling
```

```
[2dsupercooling]
type : cell
width : 100
height : 100
delay : transport
defaultDelayTime : 100
border : wrapped
neighbors :  2dsupercooling(-1,0)
2dsupercooling(0,-1)
neighbors : 2dsupercooling(0,0)
2dsupercooling(0,1) 2dsupercooling(1,0)
initialvalue : 0
localtransition : windowgrowth-rule
statevariables: value
statevalues: 0
initialvariablesvalue: 2dsupercooling.stvalues

[windowgrowth-rule]
rule : { $value } { $value := 1; } 100 { $value =0
and truecount = 1 }
rule : { $value } { $value := 1; } 100 { $value =1
}
rule : { $value } { $value := 0; } 100 { t }
```

Note that a version compatible with the original Cell-DEVS syntax is also available for this model and all the following that will be presented.

Initial State Variables file (*2dsupercooling.stvalues*)
The initial state variable file contains the coordinate of the solid seed. Here, one cell placed in the middle of the 100x100 space is used to observe free dendritic growth:

```
(49,49)=1
```

Simulations were run using the Cloud RISE and their results will be presented in section 4.

3.2. Packard Model

3.2.1. Dendritic Growth Behaviors

As mentioned in section 2, Packard's model uses growth rules related to the sum of neighboring sites. Packard considers rules, which have the property that a site value of one remains one (no melting or sublimation). The rules also depend on neighboring site values only through their sum:

$$a_i^{t+1} = f(\sigma_i^t) \quad \text{with} \quad \sigma_i^t = \sum_{\partial \in Nbrhd} a_{i+\partial}^t \qquad (1)$$

The domain of f ranges from zero to number of neighbors; f takes on values of one to zero. These rules display four types of behavior for growth from small seeds:

- No growth when the rule maps all values of σ to zero, i.e. f(σ)=0 for all σ
- Plate Structure when for instance f(σ)=1 for σ>0
- Dendritic structure with side branches when growth inhibition is added for example f(σ)=1 when σ=1
- Growth of an amorphous circular form when even more growth inhibition is added. That is the case for f(σ)=1 when σ=2.

For this paper, we will particularly focus on the dendritic structure growth since we want to observe microscopic patterns.

Apart from the previous behaviors, Packard also added growth inhibition rules that depend on the local equilibrium temperature. In addition to the presence of solid sites in the neighborhood, a liquid cell temperature (T_i) must be less than a threshold defined as (with σ:sum of solid neighbors):

$$T_{thresh} = \lambda \times \sigma_i \times (1 - \sigma_i) \qquad (2)$$

3.2.2. Cell-DEVS Formal Specification

The formal specification (that uses the Von Neumann neighborhood) is described as follows:

```
CD = < X, Y, I, S, θ, N, d, δint, δext, τ, λ, D >
X = ∅
Y = ∅
S.Phase = {0,1}
S.Temperature = {-60 - +10}
N = {(0,-1),(0,0),(0,1),(1,0),(-1,0) }
d = 100 ms
τ: N→S
S:
```

- `Phase(cell(0,0)) = 1 and Temperature(cell(0,0)) =` **`latentHeatCst`** `if Phase(cell(0,0))=0 and # of solid neighbors = 1 and cell's temperature < Tthresh (TThresh as in (2),` **`latentHeatCst`**`: constant temperature for heat release)`
- `Temperature(cell(0,0)) = Moved towards Average (Temperature(cell(0,1)), Temperature(cell(0,-1)), Temperature(cell(1,0)), Temperature(cell(-1,0)) ) by` **`DiffusionRateCst`**

The cell phase denoted by *S.Phase* is a discrete variable and can take one of these two values: 0 or 1, where 0 indicates liquid and 1 solid. The temperature *S.Temperature* is continuous but should have appropriate supercooling values (we limited the range to from -60°C to 10°C).

3.2.3. Neighborhood and Rules

Both Von Neumann and Moore Neighborhood were used to implement the model. Implemented rules include the following:

- A site will solidify only if it has one neighboring cell and its temperature is less than the local threshold. This is done using (1) and (2).
- Temperature is set to a constant high value (***latentHeatCst***) when new solid is added. Hence, heat flows and inhibits the solidification of neighboring sites. For this case, the temperature will be set to a high constant that will represent the latent heat.
- The temperature of each cell is constantly updated by moving the temperature of the cell toward the average

of the four near neighbors. A constant is used for the diffusion rate.

Different parameters, i.e. the diffusion rate, latent heat added upon solidification and the local temperature threshold (or more specifically λ) can be used to study dendritic growth for this.

3.2.4. Cell-DEVS Implementation

A 50x50 model that has a Von Neumann neighborhood is defined. The model is defined as to have periodic boundaries on the left and right sides i.e. considered in contact with each other. Two special zones are defined for the top and bottom in order for them to be insulated. For the state, the cells are all liquid at the beginning and the temperature set to the initial supercooling temperature. The crystal seeds, from which the solidification starts is in defined in the *supercooling2dext.val* file.

```
#include(Supercooling2DExtMacros.inc)

[top]
components : supercooling2D

[supercooling2D]
type : cell
dim : (50,50)
delay : transport
defaultDelayTime  : 100
border : wrapped
neighbors : supercooling2D (-1,0)
neighbors : supercooling2D (0,-1)  supercooling2D
(0,0) supercooling2D (0,1)
neighbors : supercooling2D (1,0)
zone : insulatingTop-rule { (0,0)..(0,49) }
zone : insulatingBottom-rule { (49,0)..(49,49) }
initialValue : -1
initialCellsValue : supercooling2dext.val
stateVariables : latentHeat lambda diffusionRate
initialTemperature
stateValues : -20 3 5 -36
neighborports: phase temperature
localtransition : supercooling2d-rule
```

This version uses the extended CD++ syntax, and has multiple state variables and ports. For the state variables, we have chosen *diffusionRate*, *latentHeat*, *lambda* and the initial undercooling temperature for parameters that influence the state and temperature of each cell. These were deducted from the different parameters that were cited in Packard [11] and presented previously. They can be initialized to different values in order to conduct experiments. The default values are -20ºC for the latent heat (that is the temperature is raised to -20ºC when a site solidify; this value should be greater than the initial supercooling temperature which is -36ºC by default); 3 for lambda used in (2) for the computation of the local freezing threshold temperature. For the diffusion rate, a default value of 5 is used to move the temperature of a cell towards the neighboring sites average.

Besides state variables, we define two ports: temperature (in degree Celsius) and phase (0 when liquid, and 1 for solid).

For the rule implementation, a first initialization rule sets all the ports to the right value (that is initialises the cells to the correct phase and temperature ports) since ports all have the same value: value of *initialValue*

```
[supercooling2d-rule]
%Initialization
rule : { ~phase := 0; ~temperature :=
$initialTemperature; } 100 { (0,0)~phase = -1 AND
(0,0)~temperature = -1 }
rule : { ~phase := 1; ~temperature :=
$initialTemperature; } 100 { (0,0)~phase = -2 AND
(0,0)~temperature = -2 }
```

After the initialization process, two main steps happens: phase transition for liquid cells that meet the conditions, and the heat conduction phase where the cell temperature is updated. *macro(C)* counts the sum of solid neighbors (1).

```
%Ice Propagation
% Solidification and Latent Heat Release
rule : { ~phase := 1;~temperature := $latentHeat;
} 100 { ((0,0)~phase = 0) AND ( #macro(C) = 1) AND
(#macro(temperature) < $lambda*(#macro(C)*(1 -
#macro(C))) ) }
% Heat Conduction
rule : { ~temperature :=(0,0)~temperature -
$diffusionRate; } 100 { (0,0)~temperature -
(((0,1)~temperature + (0,-1)~temperature +
(1,0)~temperature + (-1,0)~temperature)/4) >
$diffusionRate}
rule : { ~temperature :=(0,0)~temperature +
$diffusionRate; } 100 { (((0,1)~temperature + (0,-
1)~temperature + (1,0)~temperature + (-
1,0)~temperature)/4) - (0,0)~temperature >
$diffusionRate}

rule : { ~temperature :=(((0,1)~temperature + (0,-
1)~temperature + (1,0)~temperature + (-
1,0)~temperature)/4); } 100 { t }
```

```
#BeginMacro(C)
( (0,1)~phase + (0,-1)~phase + (1,0)~phase +  (-
1,0)~phase )
#EndMacro
```

For the insulated borders, that is the top and bottom edges, special rules are defined. We also have a fully insulated implementation, with non-periodic left and right sides. The insulating top rule removes the above neighbor, (-1,0), from the solid neighbors count and the temperature computation.

```
[insulatingTop-rule]
%Initialization
rule : { ~phase := 0; ~temperature :=
$initialTemperature; } 100 { (0,0)~phase = -1 AND
(0,0)~temperature = -1 }
rule : { ~phase := 1; ~temperature :=
$initialTemperature; } 100 { (0,0)~phase = -2 AND
(0,0)~temperature = -2 }
```

```
%Ice Propagation
% Solidification and Latent Heat Release
rule : { ~phase := 1;~temperature := $latentHeat;
} 100 { ((0,0)~phase = 0) AND ( #macro(CTop) = 1)
AND (#macro(temperature) <
$lambda*(#macro(CTop)*(1 - #macro(CTop))) ) }
% Heat Conduction
rule : { ~temperature := (0,0)~temperature -
$diffusionRate; } 100 { (0,0)~temperature -
(((0,1)~temperature + (0,-1)~temperature +
2*(1,0)~temperature)/4) > $diffusionRate }
rule : { ~temperature := (0,0)~temperature +
$diffusionRate;  } 100 { (((0,1)~temperature +
(0,-1)~temperature + 2*(1,0)~temperature)/4) -
(0,0)~temperature > $diffusionRate }
rule : { ~temperature := (((0,1)~temperature +
(0,-1)~temperature + 2*(1,0)~temperature)/4); }
100 { t }
```

The insulating top rule removes the below neighboring cell (1,0) from the solid neighbours count and the temperature computation.

```
[insulatingBottom-rule]
%Initialization
rule : { ~phase := 0; ~temperature :=
$initialTemperature; } 100 { (0,0)~phase = -1 AND
(0,0)~temperature = -1 }
rule : { ~phase := 1; ~temperature :=
$initialTemperature; } 100 { (0,0)~phase = -2 AND
(0,0)~temperature = -2 }

%Ice Propagation
% Solidification and Latent Heat Release
rule : { ~phase := 1;~temperature := $latentHeat;
} 100 { ((0,0)~phase = 0) AND ( #macro(CBottom) =
1) AND (#macro(temperature) <
$lambda*(#macro(CBottom)*(1 - #macro(CBottom))) )
}
% Heat Conduction
rule : { ~temperature := (0,0)~temperature -
$diffusionRate; } 100 { (0,0)~temperature -
(((0,1)~temperature + (0,-1)~temperature + 2*(-
1,0)~temperature)/4) > $diffusionRate }
rule : { ~temperature := (0,0)~temperature +
$diffusionRate;  } 100 { (((0,1)~temperature +
(0,-1)~temperature + 2*(-1,0)~temperature)/4) -
(0,0)~temperature > $diffusionRate }
rule : { ~temperature := (((0,1)~temperature +
(0,-1)~temperature + 2*(-1,0)~temperature)/4); }
100 { t }
```

3.3. Rule-Based Lattice Computer Models for Dendritic Growth Simulation

The objective of this part was to study the application of rule-based lattice modelling to simulate and understand the growth of branched dendrites. Heat transfer and solidification are simulated based on a 2D cubic lattice. Two set of simulations illustrated the application of the simulation models to the solidification of columnar and equiaxed dendritic grain structures. Initial crystal seeds are attributed different id.

3.3.1. Cell-DEVS Formal Specification

The formal specification (that uses the Von Neumann neighborhood) is described as follows:

$CD = < X, Y, I, S, \theta, N, d, \delta_{int}, \delta_{ext}, \tau, \lambda, D >$
$X = \emptyset$
$Y = \emptyset$
$S.Phase = \{0\text{-}20\}$
$S.Temperature = \{-60 - +10\}$
$N = \{(0,-1),(0,0),(0,1),(1,0),(-1,0)\}$
$d = 100$ ms
$\tau: N \rightarrow S$
S:

- Phase(cell(0,0)) = x (x is between 1 and 20) and Temperature(cell(0,0)) = cst if Phase(cell(0,0)) = 0 and # of solid neighbors = 1 (x is the Phase of that solid neighbor) and cell's temperature < < Tthresh (TThresh as in (2),**latentHeatCst**: constant temperature for heat release)
- Temperature(cell(0,0)) = Moved towards Average (Temperature(cell(0,1)), Temperature(cell(0,-1)), Temperature(cell(1,0)), Temperature(cell(-1,0))) by **DiffusionRateCst**

The cell phase denoted by *S.Phase* is a discrete variable and can take values between 0 and 20, with 0 indicating liquid and values greater than 1 solid. This is to show accurately from what seed influenced the solidification of a cell. The temperature *S.Temperature* is continuous but should have appropriate supercooling values.

3.3.2. Rules

The model basically reuses Packard concept and relies on the following rules:

1. Each seed is identified by a number, and each time a cell is solidified, it obtains same id number as the initial solid seed to track the evolution of the system.
2. Each cell temperature is adjusted using the average of the Von Neumann neighbors.
3. A cell solidifies when enough neighbours are solid.
4. The solidification is limited to those cells having lower temperature than a critical limit.
5. The critical limit is computed as the Solid-Liquid interfacial energy constant of the material divided by the local curvature. For simplification, the local curvature is computed using the count of solid neighbours.
6. When a cell solidifies the temperature is raised to a constant value.
7. No sublimation or melting is implemented by the model. A solid cell will always stay solid.

For our simulation, we limited the number of different initial solid seeds to 20.

3.3.3. Cell-DEVS Implementation

The initialization and solidification rules were changed to include the initiator seed id. The phase of each cell is always set to the same phase as the crystal seed that enabled growth. The following extract shows the main change in the solidification rule where the macro *getSeedId* identifies the original solid cell from which the dendrites will grow:

```
% Solidification and Latent Heat Release
rule : { ~phase := #macro(getSeedId); ~temperature
:= $latentHeat; } 100 { ((0,0)~phase = 0) AND (
#macro(C) = 1) AND (#macro(temperature) <
$lambda*(#macro(C)*(1 - #macro(C))) ) }
% Heat Conduction
rule : { ~temperature :=(0,0)~temperature -
$diffusionRate; } 100 { (0,0)~temperature -
(((0,1)~temperature + (0,-1)~temperature +
(1,0)~temperature + (-1,0)~temperature)/4) >
$diffusionRate}
rule : { ~temperature :=(0,0)~temperature +
$diffusionRate; } 100 { (((0,1)~temperature + (0,-
1)~temperature + (1,0)~temperature + (-
1,0)~temperature)/4) - (0,0)~temperature >
$diffusionRate}
rule : { ~temperature :=(((0,1)~temperature + (0,-
1)~temperature + (1,0)~temperature + (-
1,0)~temperature)/4); } 100 { t }
```

```
#BeginMacro(getSeedId)
( if( (0,1)~phase != 0, (0,1)~phase, if( (0,-
1)~phase != 0, (0,-1)~phase, if( (1,0)~phase != 0,
(1,0)~phase, (-1,0)~phase )  )  )  )
#EndMacro
```

3.3.3. Constrained Dendritic growth

For columnar grain simulation, the above model is used and crystal seeds are randomly located along the bottom edge. A 50x50 model, instead of 350x350 in the paper, was chosen to run the simulation. Crystal seeds are placed on the bottom edge of the square for constrained growth. For this purpose, we developed crystal seeds file generator that outputs different grain configuration.

Crystal Seed Generator

The random seed generator takes multiples parameters in order to generate the val file: the output file name, the width and height of the cell space, the initial id (the first seed id, others will be automatically generated by incrementation), the number of seeds and the dendritic growth type (free or constrained).

The following snippet show how the constrained dendritic growth seeds are generated. The X coordinate is fixed while Y is obtained by using a random generator.

```
case 1:                                    // Constrained Growth
  while (cells.size() != seeds){
    cells.clear();
    int i = height - 1;
    for (int j = 1; j < width -1; j++ ){
      if (dist(mt) < seeds * height ){
      /*  Previous mt declaration: std::mt19937 mt(rd()); */
        cells.push_back(std::pair<int, int>(i, j));
      }
    }
  }
```

VAL file

The following is a test run example:

Input: `ConstrainedSeeds.val 50 50 -4 6 1`

Output: A file named ConstrainedSeeds.val with 6 randomly located solid seeds across the bottom :

```
(49,9)  = -4
(49,13) = -5
(49,22) = -6
(49,25) = -7
(49,31) = -8
(49,36) = -9
```

3.3.4. Free/Equiaxed Dendritic growth

Cube shaped crystals were randomly located within the square to simulate equiaxed grains.

Crystal Seed Generator

The following snippet show how the constrained dendritic growth seeds are generated. X and Y seed coordinate are generated using a random generator.

```
case 0:                                    // Free Growth
  while (cells.size() != seeds){
    cells.clear();
    for (int i = 1; i < width -1; i++ ){
      for (int j = 1; j < height -1; j++ ){
        if (dist(mt) < seeds ){
       /*  Previous mt declaration: std::mt19937 mt(rd()); */
          cells.push_back(std::pair<int, int>(i, j));
        }
      }
    }
  }
```

VAL file

The following is a test run example:

Input: `FreeSeeds.val 50 50 -2 5 0`

Output – FreeSeeds.val:

```
(5,12)  = -2
(12,31) = -3
(18,24) = -4
(21,7)  = -5
(24,10) = -6
```

3.4. Other Models

We will mainly discuss the model described in [12] that suggests a new dendritic growth cellular automata model.

3.4.1. New CA based on Von-Neumann/Moore Neighbors

The formal specification is described as follows:

```
CD = < X, Y, I, S, θ, N, d, δint, δext, τ, λ, D >
X = Ø
Y = Ø
S= [0,1]
N = Moore neighborhood
d = 100 ms
τ: N→S
S:  (using beta=0.00001, and alpha=1)
```

- `1 if (0,0) >= 0.98 % solid cell`
- `(0,0) + 0.00001 + ((1/16)*((-8*(0,0)) + liquidConcentration((-1,-1),(-1,0),(-1,1),(0,-1),(0,1),(1,-1),(1,0),(1,1))`

- `(0,0) + ((1/16)*((-8*(0,0)) + liquidConcentration( (-1,-1) + (-1,0) + (-1,1) + (0,-1) + (0,1) + (1,-1) + (1,0) + (1,1))`

Where *liquidConcentration* returns the sum of the concentration of only cells that are liquid in the parameter set(Moore neighborhood here).

The Von Neumann uses the same concept but with different fractions: 1/8 is used instead of 1/16, and -4 instead of -8 in both the liquid and near-solid diffusion rules.

The following snippet shows the corresponding advanced Cell-DEVS model for the Von Neumann neighborhood this time:

```
[2dsupercooling]
type : cell
width : 100
height : 100
delay : transport
defaultDelayTime : 1
border : wrapped
neighbors : 2dsupercooling(-1,0)
neighbors : 2dsupercooling(0,-1)
2dsupercooling(0,0)   2dsupercooling(0,1)
neighbors : 2dsupercooling(1,0)
initialValue : -1
initialCellsValue : 2dsupercooling.val
localtransition : 2dsupercooling-rule
neighborports: value phase

% initial value 0.3, state 0
[2dsupercooling-rule]
% Initialization
rule : { ~value := 0.3; ~phase := 0; } 1 {
(0,0)~value = -1 AND (0,0)~phase = -1 }
rule : { ~value := 1.0; ~phase := 2; } 1 {
(0,0)~value = -2 AND (0,0)~phase = -2 }

% Solidifcation Rule
rule : { ~value := 1; ~phase := 2; } 1 {
#macro(isSolid) }
rule : { ~value := (0,0)~value +
#macro(diffusionNearSolid) ; ~phase := 1; } 1 {
#macro(isNearSolid) }
rule : { ~value := (0,0)~value +
#macro(diffusionSolution) ; ~phase := 0;} 1 {
#macro(isSolution) }
```

In this implementation, a phase of 0 represents liquid/solution, 1 is for near-solid and 2 indicates a solid cell. The value of each cell reflects the concentration. For the rules, a diffusion term is added to liquid and near-solid cells while solid cells remain solid.

A snippet containing of macros is shown below:

```
#BeginMacro(isSolution)
((0,0)~value < 1) AND ( ((-1,0)~value < 1) AND
((0,-1)~value < 1) AND  ((0,1)~value < 1)  AND
((1,0)~value < 1) )
#EndMacro

#BeginMacro(isNearSolid)
((0,0)~value < 1) AND ( ((-1,0)~value >= 1) OR
((0,-1)~value >= 1) OR  ((0,1)~value >= 1)  OR
((1,0)~value >= 1) )
#EndMacro
#BeginMacro(diffusionSolution)
(1/8)*( (-4*(0,0)~value) + if((-1,0)~phase = 0, (-1,0)~value,0) + if((0,-1)~phase = 0, (0,-1)~value,0) + if((1,0)~phase = 0, (1,0)~value,0) + if((0,1)~phase = 0, (0,1)~value,0) )
#EndMacro

#BeginMacro(diffusionNearSolid)
0.0001 + ( (1/8)*( (-4*(0,0)~value) + if((-1,0)~phase = 0, (-1,0)~value,0) + if((0,-1)~phase = 0, (0,-1)~value,0) + if((1,0)~phase = 0, (1,0)~value,0) + if((0,1)~phase = 0, (0,1)~value,0) ) )
#EndMacro
```

We run these simulations using the fastest RISE server since these were quite heavy. It is also essential to note that the author might have provided an erroneous rule since small experiments did not show any dendritic structure.

3.4.2 Other implementations

We also evaluated the possibility of implementing the models proposed in [13] and [14], but time constraints did not allow us to complete them.

4. SIMULATION RESULTS

4.1. Window Automata and Dendritic Growth

The following figure shows the expected result and shown in the reference paper.

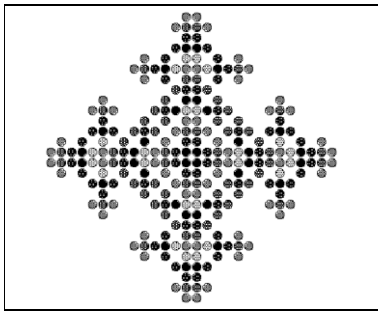
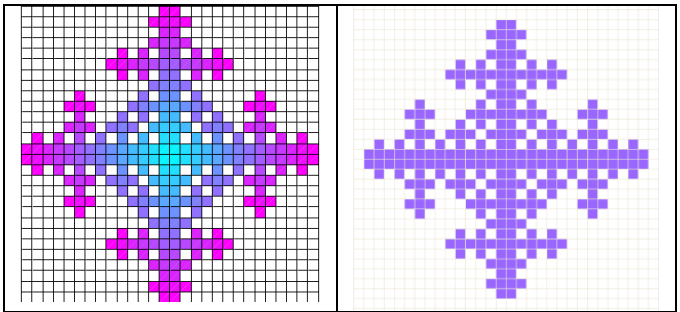


Figure 1. Theoritical Results [10]

In the following figure, the rightmost figure was produced by the drawlog tool and visualized with CD++ Modeler and is identical to the leftmost figures that was implemented using CA.

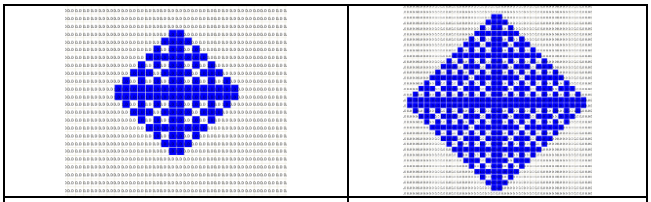


Cellular Automata [10]	Cell-DEVS

Figure 2. Simulation Results

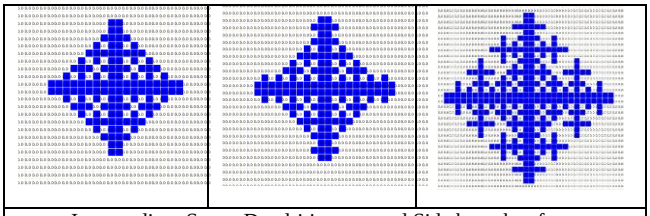
For this implementation, colored (blue/purple) cells have a value of 1. The simulation is initiated with a 2x2 seed placed at the center of the square. Then, only cells with one solid neighbor become solid.

An essential observation that was made is that every 2n time steps, the growing seed forms a plate structure, then dendritic arms grow from the corners of the plate, side branches form, and again all side branches grow into a plate and the process is repeated.



Starting Plate Structure	Plate Structure after 2n steps

Figure 3. Plate Structures



Intermediary Steps: Dendritic arms and Side branches form

Figure 4. Dendritic Arms Growth

Complex shapes can also be generated using the window automata rule. The Moore neighborhood was tested as well and results are enclosed in the files attached to this paper.

4.2. Packard Model

The following palette is used for all the following experiment to identify thermal dendrite; temperature ranges are specified on the left side, and the associated color is shown on the right:

From	To	Color
-36	-36,001	
-36,001	-28	
-28	-23	
-23	-18	
-18	-13	
-13	-8	
-8	-3	
-3	2	
2	7	
7	2	

Figure 5. Temperature Palette used for Dendrites Identification

4.2.1. Growth Behaviors

As mentioned earlier, multiple growth behavior can be generated by mapping the solidification function to different values of the sum of solid neighbors. We were able to observe all of them. We use the Moore variant for the result shown in Figure 6 since the Von Neumann was presented for the previous results. Note that dark blue shows liquid cells at the initial undercooling temperature. For the following experiments, the seed is 3x3.

From the results shown in Figure 6, we can conclude that the number of solid neighbors should not be too high or too low in order to observe dendritic pattern.

4.2.2. Parameter Variation

a) Effect of Lambda

When lambda is varied, we notice that the structure goes from tendril growth to a stable dendritic structure. The Figure 7 shows the obtained results on the left. The right shows results obtained by Packard. In this later case however, a different neighborhood is used. When λ is small Tthresh is lower so that more solidification occurs. In addition, more heat diffuses before a boundary site solidifies. That is why the sites have different colors, i.e. different temperatures since latent heat is only released when the site becomes solid. The solidification is inhibited by higher λ since they results in an increased threshold temperature.

b) Latent Heat Effect

The latent heat parameter determines how warm a newly solidified cell becomes as to represent the energy liberated when solidification occurs. The higher the latent heat, the less neighbor cells solidify. This is because the temperature at the solid-liquid interface increases and inhibits the freezing process. Hence, the temperature of the cell will tend to be higher than TThresh, the Threshold temperature.

Besides, dendrites that form have higher temperature and are warmer (see palette colors for T=0) compared to lower latent heat temperature (T=-20). These results are shown in Figure 8 (shown in a subsequent page),

c) Diffusion Rate Effect

The diffusion rate parameter is the constant that is added or subtracted in order to move a cell temperature towards the average of the neighboring cells. When this constant is high, the cell gets close to the neighboring temperature faster and the thermal dendrites will tend to be warmer. This will also highly impact the growth behavior and structure of the solution as illustrated in Figure 9.

d) Combination of factors

When different parameters are changed, we notice that some transitions happen. In Figure 10 (shown in the next pages) for instance, lowering the latent and diffusion rate made a tendril-like growth become amorphous.

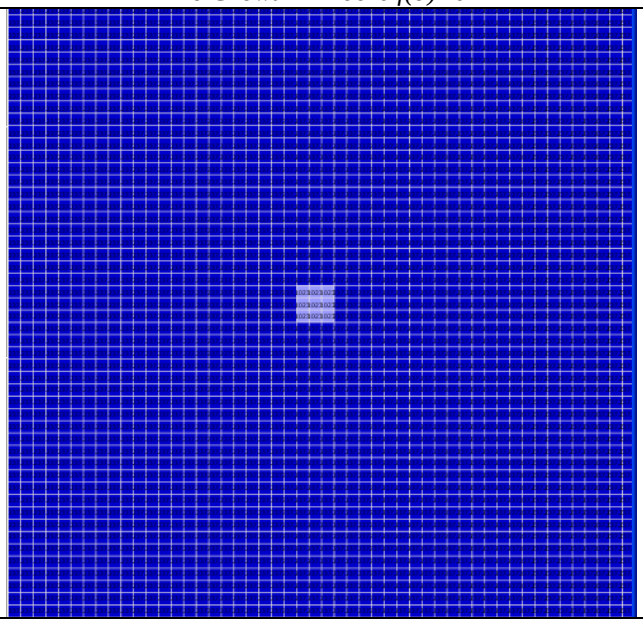
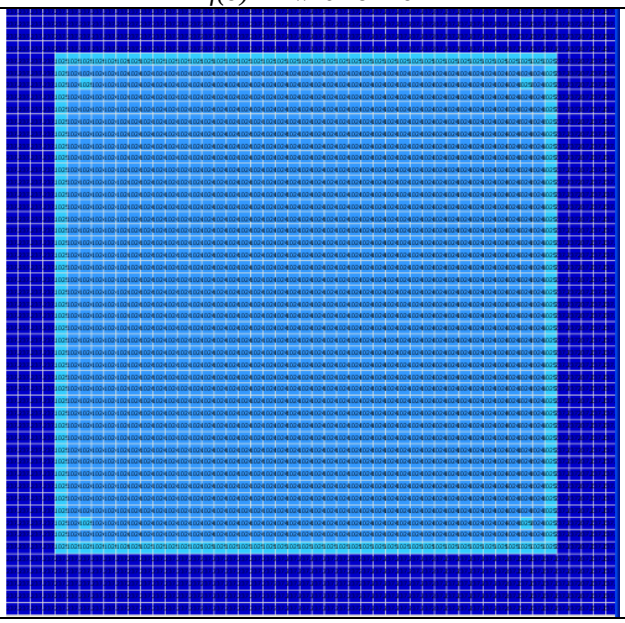
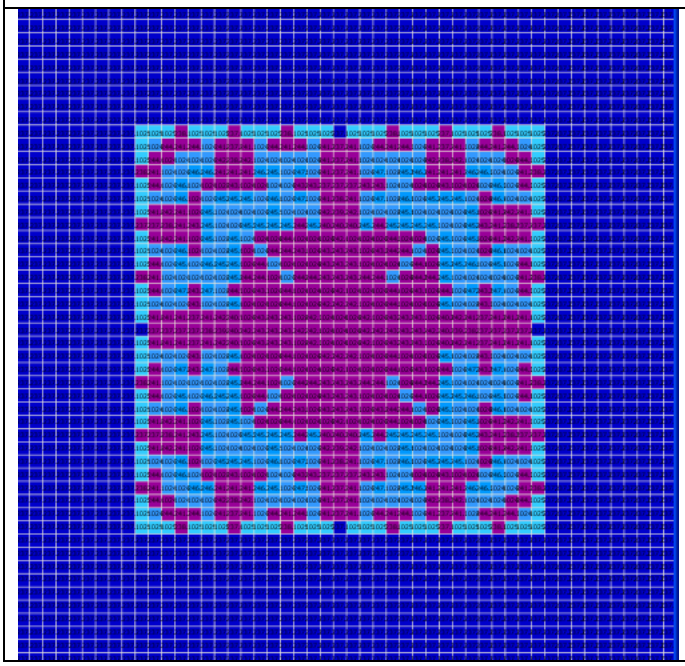
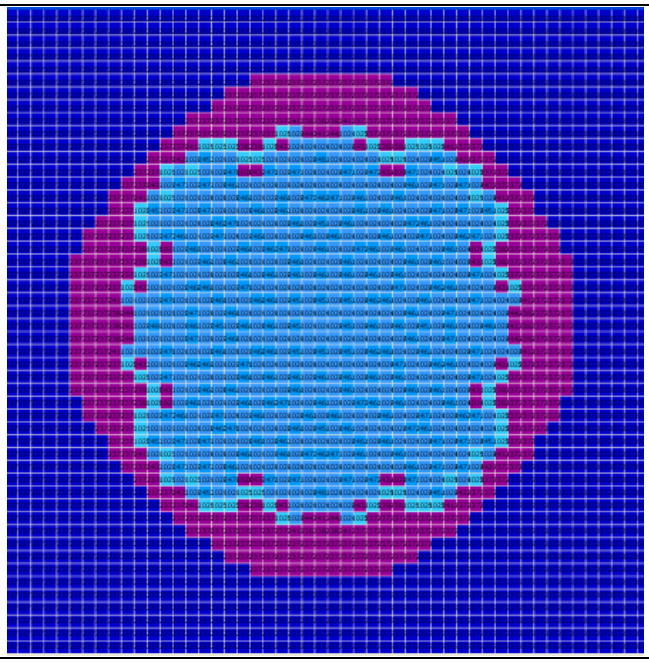
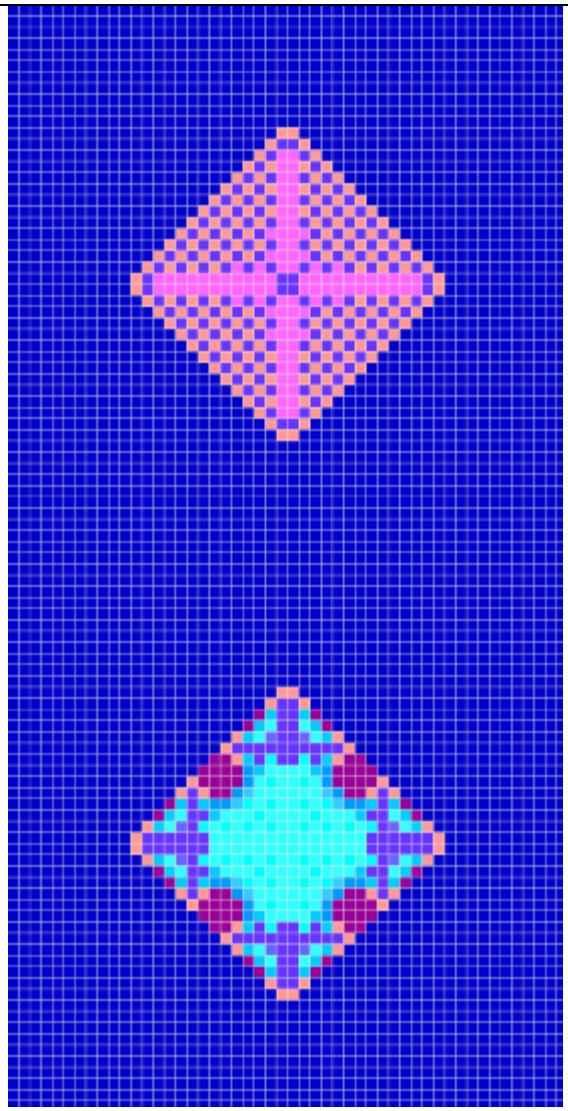
No Growth – Moore f(σ)=0	*Plate structure reflecting the lattice structure – Moore: f(σ)= 1 when σ > 0*
	
Dendritic structure with side branches when growth inhibition is added for example f(σ)=1 when σ=1	*An amorphous, asymptotically circular form – Moore: f(σ)= 1 when σ = 2*
	

Figure 6. Types of Growth Behaviors

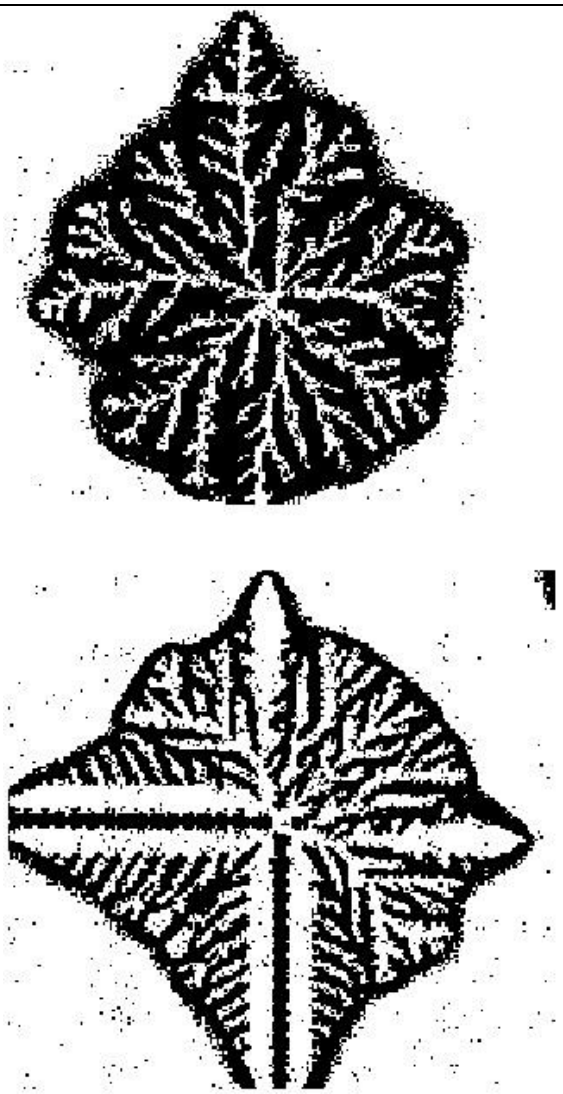
From	To	Color
-36	-36,001	
-36,001	-28	
-28	-23	
-23	-18	
-18	-13	
-13	-8	
-8	-3	
-3	2	
2	7	
7	2	

The palette shown on the left side (temperature range and corresponding color) can serve as a guide to visualize the results above: Dark blue is liquid cell at the initial temperature (-36ºC), Dark purple are at the interface of the solidification. The number of solid neighbors affects greatly the final structure. An analysis has been done in 4.2.1.and can be read for reference.

Lambda Effect



Cell-DEVS Results (λ=0.01 on Top vs λ=100 on the bottom)



Packard Results(Low λ on top vs high λ on bottom)

Figure 7. Lambda Variation Effect (Low lambda on the upper figures and low lambda on the lower figures)

From	To	Color
-36	-36,001	
-36,001	-28	
-28	-23	
-23	-18	
-18	-13	
-13	-8	
-8	-3	
-3	2	
2	7	
7	2	

The palette shown on the left side (temperature range and corresponding color) can serve as a guide to visualize the results above: Dark blue is liquid cell at the initial temperature(-36ºC), Dark purple are at the interface of the solidification. λ is low for the top figures and high for the bottom figures. In the latter case, we observe a clear anisotropy and stable branching. Note that we are using $f(\sigma)= 1$ when $\sigma >= 1$ for this experiment. An analysis has been done in ***4.2.2..a)****and can be read for reference. t*

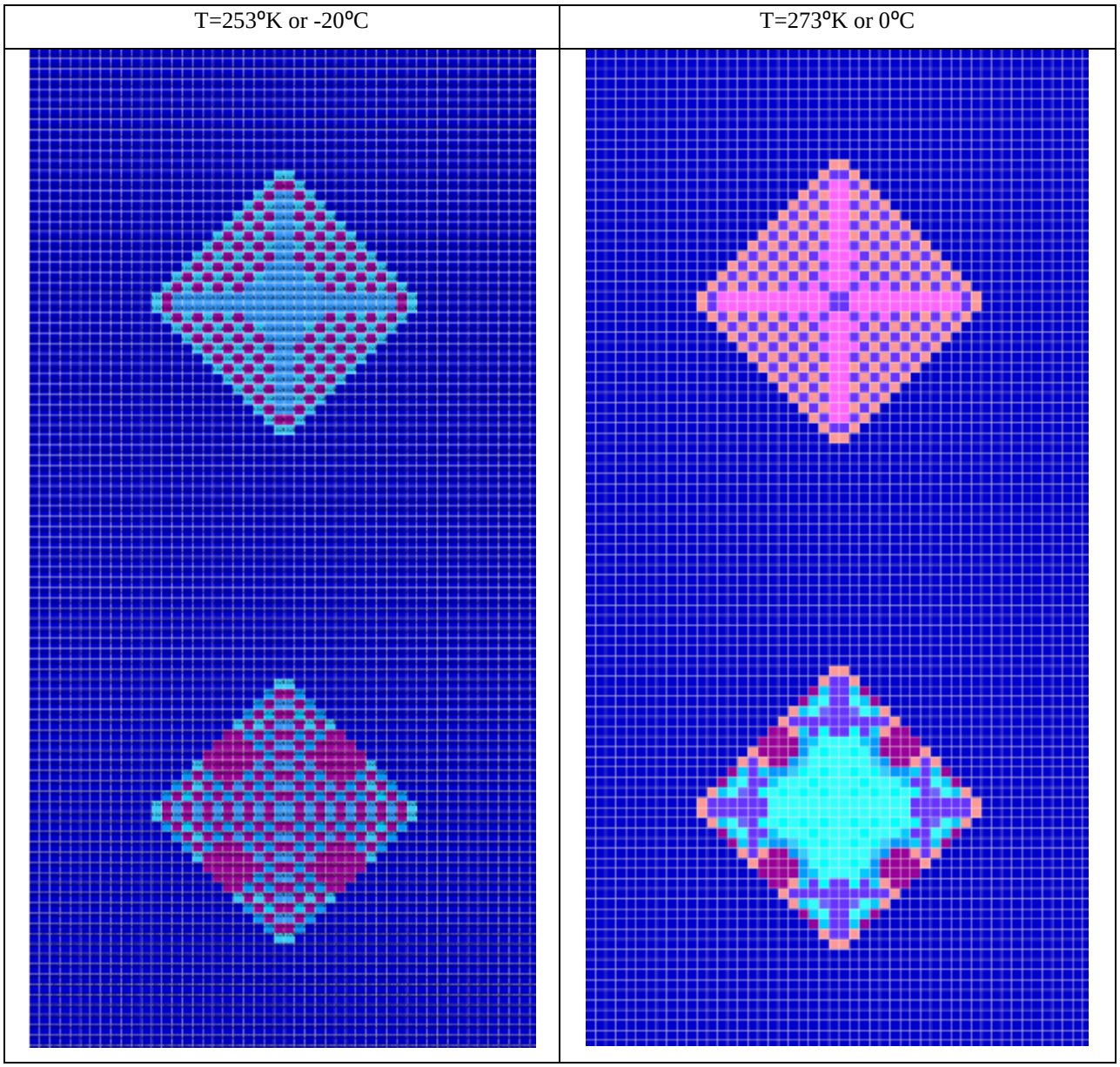


Figure 8. Latent Heat Variation (Low Latent Heat on Left and High Latent Heat on the right - Lambda is as described in Figure 7)

From	To	Color
-36	-36,001	
-36,001	-28	
-28	-23	
-23	-18	
-18	-13	
-13	-8	
-8	-3	
-3	2	
2	7	
7	2	

The palette shown on the left side (temperature range and corresponding color) can serve as a guide to visualize the results above: Dark blue is liquid cell at the initial temperature (-36°C), Dark purple are at the interface of the solidification. λ is low for the top figures and high for the bottom figures. In the latter case, we observe a clear anisotropy and stable branching. Note that we are using $f(\sigma)= 1$ when $\sigma >= 1$ for this experiment. An analysis has been done in ***4.2.2..b).*** *and can be read for reference.*

Diffusion Rate Effect

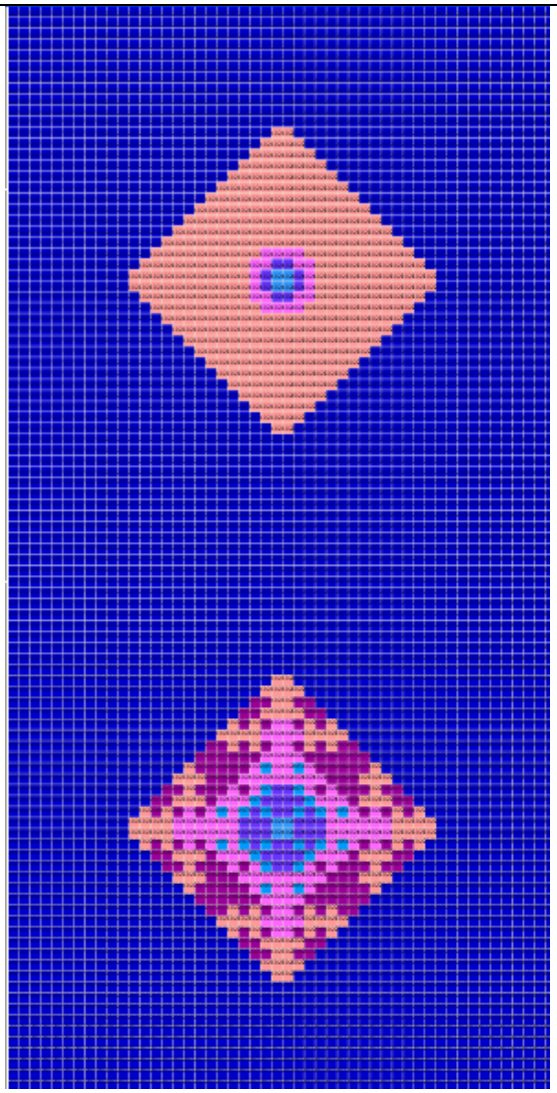
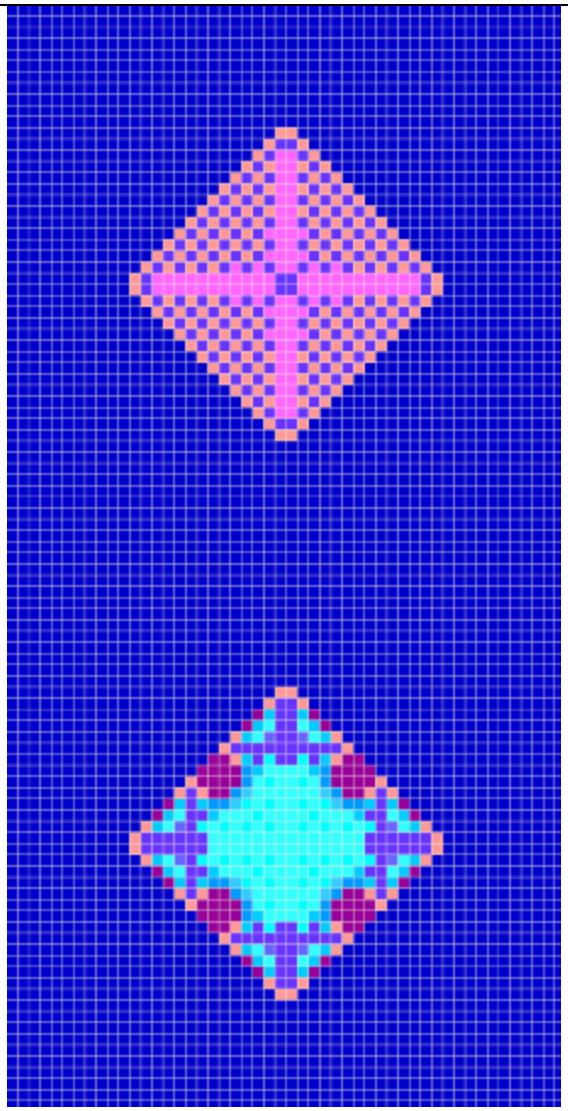
Constant Diffusion Rate = 1	Constant Diffusion Rate = 10
	

Figure 9. Diffusion Rate Constant Effect (Low rate on Left and High rate on the right - Lambda as described in Figure 7)

From	To	Color
-36	-36,001	
-36,001	-28	
-28	-23	
-23	-18	
-18	-13	
-13	-8	
-8	-3	
-3	2	
2	7	
7	2	

The palette shown on the left side (temperature range and corresponding color) can serve as a guide to visualize the results above: Dark blue is liquid cell at the initial temperature (-36ºC), Dark purple are at the interface of the solidification. λ is low for the top figures and high for the bottom figures. An analysis has been done in ***4.2.2.c).*** *and can be read for reference. t*

Combination of factors

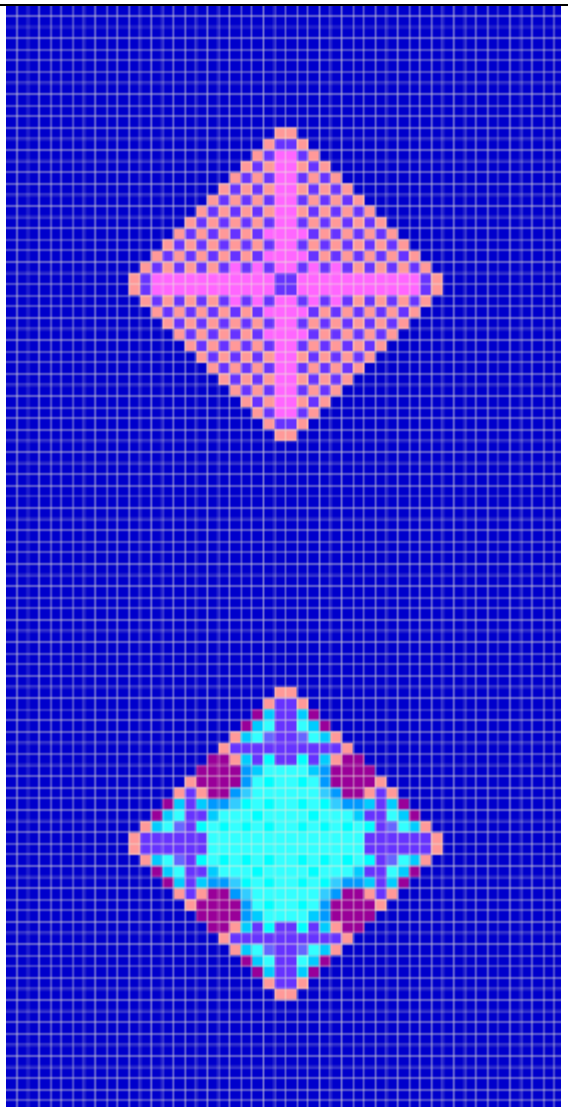
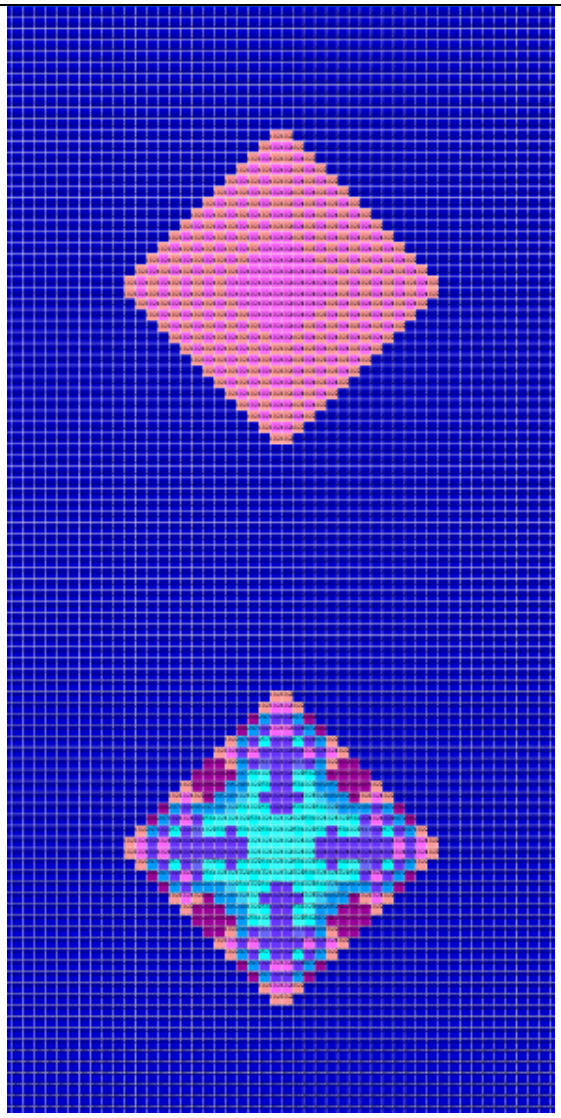
T=273°K=0°C, Same Lambda, Diff = 10	T=253°K=-20°C, Same Lambda, Diff = 5
	

Figure 10. Multiple Factors Variation (Latent Heat=0°C & Diffusion Rate=10 VS Latent Heat=-20°C & Diffusion Rate=10)

For figure 10, the palette is the same as in all the previous cases: Dark blue is liquid cell at the initial temperature (-36ºC), Dark purple are at the interface of the solidification. λ is low for the top figures and high for the bottom figures.

*An analysis has been done in **4.2.2..d).** and can be read for reference*

*For figure 11, solid is dark blue, and liquid is light blue. See **4.5.** for the analysis.*

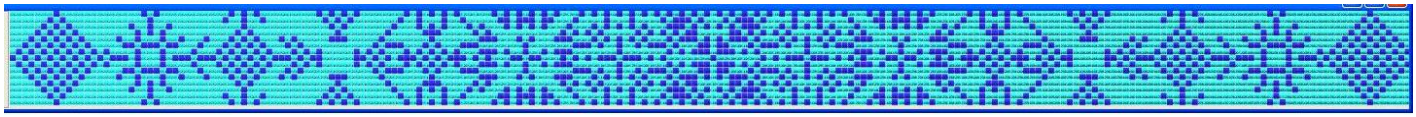


Figure 11. A 3D Simulation Result that uses the Window Automata principle

4.3. Rule-Based Lattice Computer Models for Dendritic Growth Simulation

Two set of simulations are presented to show the application of the model to the solidification of columnar and equiaxed dendritic grain structure. For the following experiment results, all the cells have the same initial supercooling temperature, -36ºC. The growth is initiated from 2x2 seeds randomly place at the bottom edge for the constrained growth, and randomly dispersed all over the cell space for the free dendritic growth case.

For visualizing results, we use a palette that assign a specific color by id. The numbers show the seed id, and the assigned color is on the right side.

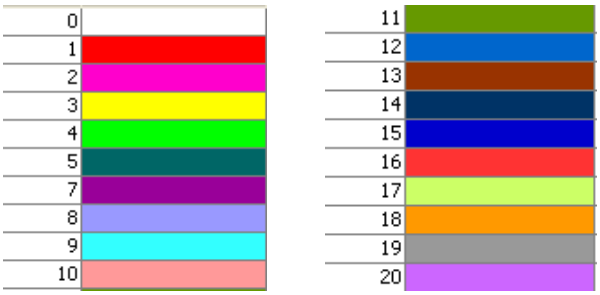


Figure 12. Palette used for Dendrites Id

Hence, we can easily see from which seed, the dendrites spread. Note that for thermal visualization, the Packard palette is used.

4.3.1. Constrained Growth

Constrained growth lead to columnar dendrites since the neighboring dendrites grown from different seeds inhibit horizontal growth once dendrites collide. Indeed the temperature is raised, and no solidification occurs.

The following capture shows thermal dendrites results obtained by Brown.

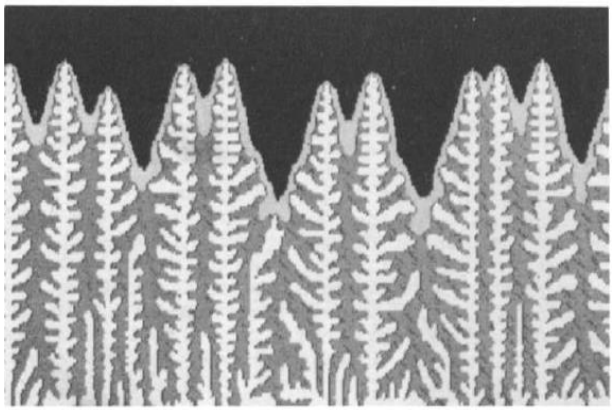


Figure 13. Reference Paper Results [9]

With our Cell-DEVS implementation, we obtain similar behavior and a similar structure.

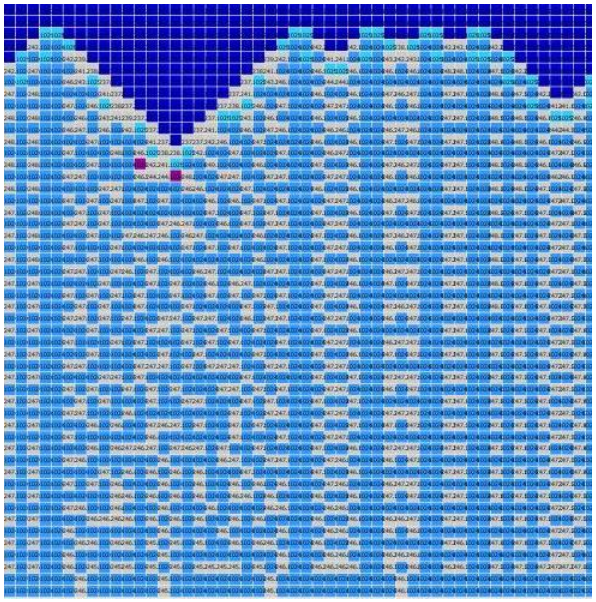


Figure 14. Cell-DEVS Columnar Thermal Dendrites

When we add, the identification component, we can easily follow how dendritic growth progresses. At the start:

Figure 15. Constrained Dendrites Simulation (Start)

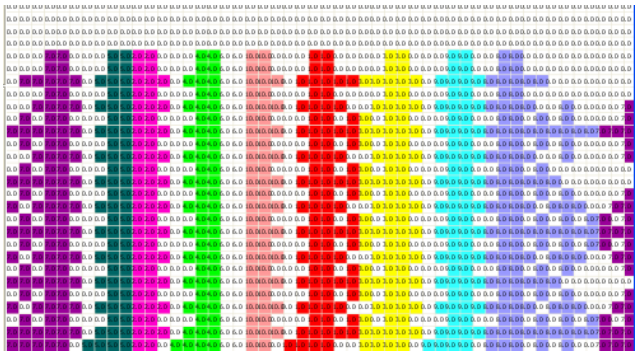


Figure 16. Constrained Dendrites Simulation (22 steps later)

4.3.2. *Free Growth*

Dendrites grow in all crystallographic directions until they hit dendrites issued from a different seed.

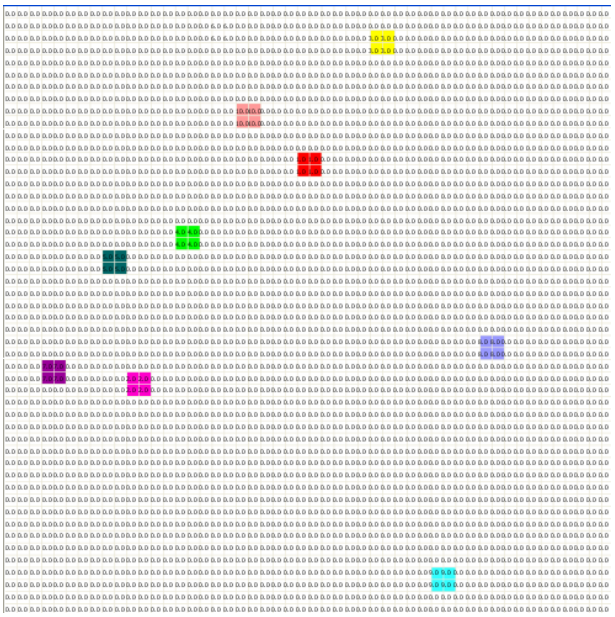


Figure 17. Free Dendrites Simulation (Start)

After 22 simulation steps, the dendrites have spread in all directions.

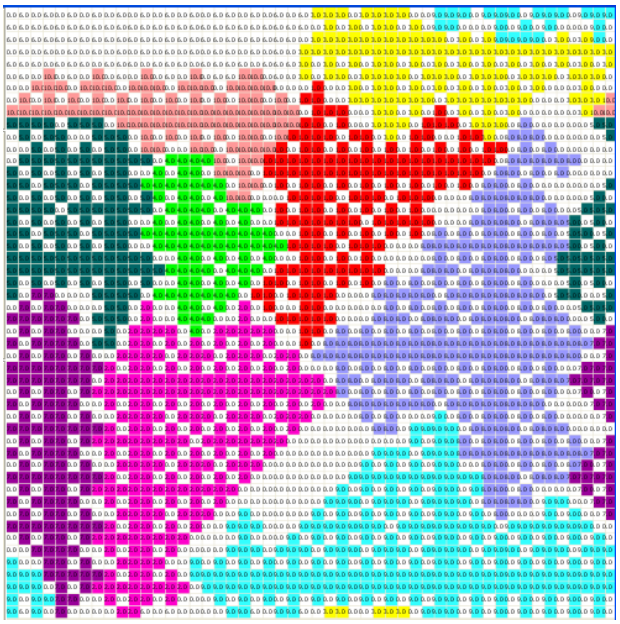


Figure 18. Columnar Dendrites Simulation (22 steps later)

We can also notice that results are similar to those in [9], except that since dendrites sometimes grow on different direction, they might be using hexagonal cells or introducing non-deterministic behaviors. However, the authors do not give any details about the specifics of their implementation but only provide a general description.

We noticed that the extended Cell-DEVS syntax did not allow the defined zone to behave as specified in the border rules. The results show that instead all borders were considered periodic. The same insulating concepts were tested in the old simulator and performed accurately. We did not investigate this issue further since it has already been reported before.

4.4. Other models

Apart from the above models, we also tried to implement another cellular automata described in [12] in order to observe finer dendrites growth. This implementation suggests a new technique based on the cell concentration, has three phases (liquid, solid and near solid) and uses the Von Neumman and Moore neighborhood. Small values are used to get results that are more precise as shown below but, the model requires heavy computations.

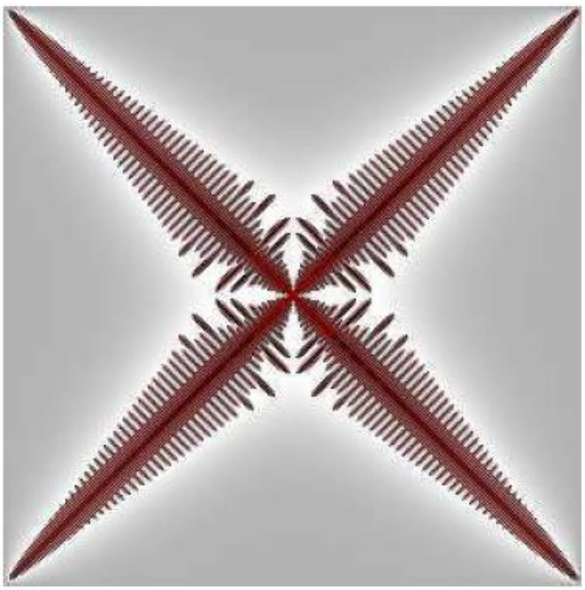


Figure 19. Expected Result for Alpha=1, Beta=0.0001 [12]

However, the simulation takes more than a day and we were not able to gather all the results in time. One of the simulation is still running here. Besides, we made certain assumptions since the paper was not specific enough: the neighborhood defined in the diffusion formula had an undefined index and a typo for the initial value. We assumed that it meant all neighbors. The experiment was run using the same values shown in Figure 19. with an initial value of 0.03.

The first tests lead to different results that do not exhibit the expected dendrites forms. Figure 20 shows the results obtained after 2000 simulation steps of a 300x300 cell space with a solid seed a placed in the center. The simulation took several hours and even when the

concentration was raised, there were no new solid cells, although near-solid cells(shown in dark blue) appeared

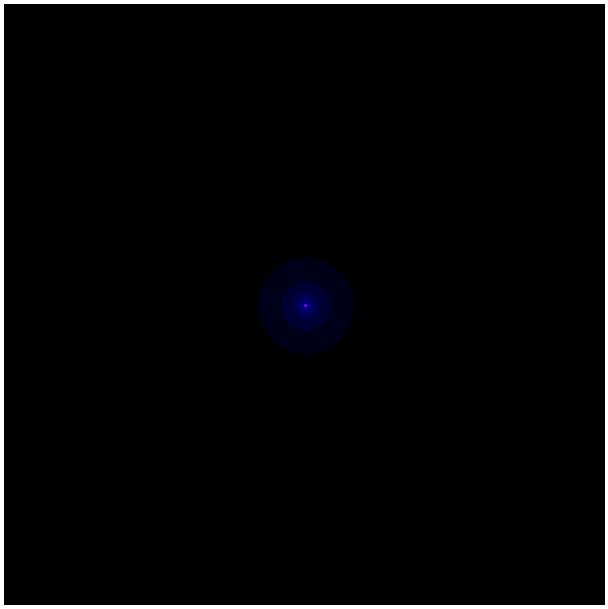


Figure 20. First Tests Results

In the above figure, an RGB is used and maps R to solid/liquid (255 for liquid and 0 for solif), and B to the concentration. Therefore, black shows liquid cells, dark blue for near-solid cells and purple for solid cells (The light purple dot in the middle is the initial solid cell).

Improved versions are still running on the server. The objective was to refine the observed dendritic patterns from the Packard and Brown models by adding an additional phase state and having a more polished solid-liquid interfaces that also takes into account mushy cells. We also wanted to add the temperature component introduced by Packard in order to have a simple Cell-DEVS model were both thermal and phase dendrites could be observed.

Indeed, with the first elementary window automata only phase/state dendrites are shown since no temperature is considered. Packard adds the temperature key ingredient that affects dendrites but does not provide multiple cell states

4.5. 3D Results with Window Automata

We have also run a simulation of a 3D model using the above criteria and observed different patterns. Figure 11 shows a -35^0C supercooled model of size 15x15x15.

Different structures are observed in different planes, and can be a mixture of amorphous, tendril-like and dendritic forms. We did not investigate this further due to time constraints. However, it would be interesting to study these patterns in a 3D environment in future work.

5. CONCLUSION

We described several rule-based lattice models for dendritic growth in supercooled liquids and presented their implementation with Cell-DEVS. Results were compared to those shown in the reference papers. We started with the Window Automata originally presented by Weisbuch [10] and that uses simple rules for elegant dendritic growth simulation. Then, the Packard model, that involves several parameters and combines both continuous and discrete elements, was discussed. Multiple behaviors similar to those observed in the original paper were observed, and the impact of additional different parameters analyzed. After that, we studied the application of the previous model both free and constrained growth by introducing multiple seeds with distinct identification.

We also studied other models to refine the observed dendrites but were not able to complete the required simulation due to time limitations and missing parameters in the paper that required various assumptions. One of the most challenging aspects of this project was to reproduce experiments since parameters were not given in most papers. In some cases, crucial information was missing. We believe that providing clear information for experiment reproducibility for modelling articles is essential, especially for such a subject that still fascinates scientists and heavy computation experts. It was quite complicates to reproduced the experiments although the rules were there. Nevertheless, multiple parameters were tested until similar results were obtained.

The other roadblock was that complementary tools provided with CD++ were not able to process some of the large models that had up to several GB of logs. Sometimes, it took days to get an error message because of out of memory exceptions for example. We acted upon these issues by reducing the size of our models and also developing special parsing and visualization tools.

Overall, we were able to observe microstructures in addition to the macroscopic features that were explore in our last assignment. We also compared the simulation time of our previous models when using RISE and the difference was remarkable in general (minutes against days). The refined model that required heavy computation appears to run longer using the RISE server, supposedly because it has multiple ports while it completes in a day using the original CD++.

For future work, the defined models could be adapted to 3D and patterns studied. Other complex behaviors such as metal alloys and crystallographic variation [14] should be introduced and studied as well. For the tool aspect, the visualization tool needs better support for larger files.

Plus, the CD++ simulation log tool should be improved to improve a current design flaw that accumulates logs in memory and only writes to the file once the simulation is over. The Lopez version has a fix for the log but does not generate the equivalent partial drawlog file.

6. REFERENCES

[1] A. S. Network, "ASN Aircraft Accident ATR-72-212," Aviation Safety Network, [Online]. Available: http://aviation-safety.net/database/record.php?id=19941031-1. [Accessed 23 10 2014].

[2] (2012, Dec.) Lost - The Mystery Of Flight 447 [Air France Flight 447]. [Online]. https://www.youtube.com/watch?v=peHo0ajeTec

[3] K. Monzen, T. Hosoda, D. Hayashi, Y. Imai, Y. Okawa, T. Kohro, H. Uozaki, T. Nishiyama, M. Fukayama and R. Nagai, "The use of a supercooling refrigerator improves the preservation of organ grafts," Biochemical and biophysical research communications, vol. 337, no. 2, pp. 534-539, 2005.

[4] I. C. London, "Supercooling explained," Imperial College London, 12 3 2012. [Online]. Available: http://wwwf.imperial.ac.uk/blog/reporter/2012/03/12/supercooling-explained/. [Accessed 23 10 2014].

[5] A. L. Greer, "Materials science: A cloak of liquidity," Nature, vol. 464, pp. 1137-1138, 2010.

[6] S. Brown and N. Bruce, "A 3-dimensional cellular automaton model of ‘free’dendritic growth," Scripta metallurgica et materialia}, vol. 32, no. 2, pp. 241-246, 1995.

[7] S.-C. Huang and M. Glicksman, "Overview 12: Fundamentals of dendritic solidification—I. Steady-state tip growth," Acta Metallurgica, vol. 29, no. 5, pp. 701-715, 1981.

[8] D. A. a. W. G. A. Rodriguez, "New Extensions to the CD++ tool," in Society for Computer Simulation International, 1998.

[9] S. Brown and J. Spittle, "Rule-based lattice computer models for simulating dendritic growth," Scripta metallurgica et materialia, vol. 27, no. 11, pp. 1599-1603, 1992.

[10] G. Weisbuch and S. Ryckebusch, Complex systems dynamics: An introduction to automata networks, Redwood City: Addison-Wesley, 1991.

[11] N. H. Packard, "Lattice models for solidification and aggregation," First International Symosium for Science on Form, pp. 95-101, 1986.

[12] Y. Zhao, S. Billing and D. Coca, "A cellular automata modelling of dendritic crystal growth based on Moore and Von Neumann neighborhood," Automatic Control and Systems Engineering, University of Sheffield, Sheffield, 2008.

[13] M. Zhu and D. Stefanescu, "Virtual front tracking model for the quantitative modeling of dendritic growth in solidification of alloys," Acta Materialia, vol. 55, no. 5, pp. 1741-1755, 2007.

[14] L. Wei, X. Lin, M. Wang and W. Huang, "A cellular automaton model for the solidification of a pure substance," Applied Physics A, vol. 103, no. 1, pp. 123-133, 2011.

[15] A. Lopez and G. Wainer, "Improved Cell-DEVS model definition in CD++," Cellular Automata, pp. 803-812, 2004