Self Organizing Particles

- Mix of biology and Computer Science
- Heavily reliant on distributed algorithms
- Many different types:
  - Active VS Passive Systems
  - Self-Organizing Networks
  - Modular Self-Reconfigurable Robotic Systems
- Hardware?
Preliminary Information

- All particles in the system can be modeled as finite state machines, with constant size memory
- Number of states is on the order of $\log n$
- Based on the Nubot Model
Model

- The space is modeled by a *Equilateral Triangular Graph*, referred to as $G_{eqt}$.

- A particle occupies either a single node or pair.
- Every node can be occupied by at most a single particle
- Particles are said to be Connected if they are adjacent to each other, they are then also referred to as neighbours.
- Since every particle is modeled by a FSM
  - each particle has a state $s$, where $s \in Q$, where $Q$ is a finite set of states
  - each particle holds a flag $f$, where $f \in \Sigma$, $\Sigma$ is an alphabet, for every edge that is incident to it.
- Particles move via Expansion and Contraction
- The edges incident to a particle are labeled as seen in the image
- Modeled using the distributed principle of locality, the knowledge of a particle is restricted to a specific ‘view’
- The system progresses using atomic actions
Object Coating is part of the set of problems called Morphing Problems.

- Morphing Problems are defined as: A system of particles has to morph into a shape with specific characteristics while sustaining connectivity.

Morphing problems can be modeled mathematically as follows: \( M = (I, G) \)

- Where \( I \) is the set of Initial graph configurations, and \( G \) is the set of Goal configurations.
In the Infinite Object Coating problem, the object has an infinite surface and, accordingly, a uniform coating is accomplished when all the particles of a system are directly connected to the object.

- We will define the problem to be as follows: Given two types of particles, object and non-object particles, the problem is complete when the non-object particles cover the object particles.
Algorithm

Intuitively: We want the particles to move, in uniform, across the specified object until the entire object has been coated. We can see this as painting a surface. We want the non-object particles to paint the entire surface of the object.
Algorithm

There are then some rudimentary operations that the particle system has to implement:

- Individual Movement
- Group Movement, this is needed to satisfy connectivity
- Some sort of Termination
The algorithms Given:

Algorithm 1: *Movement along a surface*

let $k$ be the size of the neighborhood of the particle (i.e., $k = 6$ or $k = 10$)
let $i$ be the label of an edge connected to the object

while edge $i$ is connected to the object do
    $i \leftarrow (i - 1) \mod k$

return $i$
Algorithm 2: *Spanning Forest*

(it's really long so it's on the next slide)
span-forest
; done := false
; if inactive and not done
;   if connected-to-surface?
;     phase := leader
;     movement := idle
;     done := true
;   if filter neighbours with phase leader or follower not empty
;     direction := toward first of the filter statement above
;     phase := follower
;     done := true
; if follower and not done
;   if movement is contracted and connected-to-surface?
;     phase := leader
;     movement := idle
;     done := true
;   if movement is contracted and direction is expanded
;     expand to direction (i.e. attempt-handover)
;     direction := direction of direction
;   if movement is expanded
;     if any neighbours of tail are followers
;       move := handoverContraction
;     if no neighbours of tail are followers and no inactive neighbours
;       move := contract
; if leader and not done
;   if contracted
;     move := expand
;     direction := next-move
;   if expanded
;     if any neighbours of tail ar followers
;       move := handoverContraction
;     if no neighbours of tail are followers and no inactive neighbours
;       move := contract
Algorithm

The termination algorithm:
Algorithm 3: *Complaining*

```plaintext
complain
 ; if leader
 ;   s := next-move
 ; if contracted and s = {} and any neighbours are followers or complaining
 ;   send 'complaint' -> s
 ; idle
 ; endif
 ; if contracted and no neighbours are complaining
 ; idle
 ; end
 ; endif
 ; do spanning-forest
```
Analysis

- Correctness
  - Maintains Connectivity
- Terminates
Requires Optimal Work

- The worst case work is $\Omega(n^2)$
  - See the worst case

- The particle labeled $i$ requires at least $2i$ movements before it lies contracted on the surface. Therefore, any algorithm requires at least $\sum 2i = \Omega(n^2)$
Requires Optimal Work

Three possible phases that any particle can be in:

- **inactive**
  - nothing is done here.

- **follower**
  - by connectivity, the path is bounded by $2n$. Therefore the number of movements is on the order of $n$.

- **Leader**
  - only expands if it sees some indicator. It can see at most $n-1$ indicators before termination, if we consider the final contraction to terminate the number of moves performed is on the order of $n$.

Therefore $\Omega(n^2)$, optimal work.