CS 6380
Multiagent Systems

Spring 2020

Tom Henderson

Lecture Notes 1

Multiagent Systems

- Agents
- Autonomy
- Intelligence
- Cooperative/Competive (Communication)
- Framework: Agent-Based Modeling and Simulation (ABMS)

- Project: Urban Air Mobility

Reaction-Diffusion
Cell-Level Agents

https://www.youtube.com/watch?v=alH3yc6tX98

Flocking Agents

http://www.globalstewards.org/murmurations.htm

and, Hollywood!

https://www.youtube.com/watch?v=FrqGY6sJpck

We’ll study: What is?

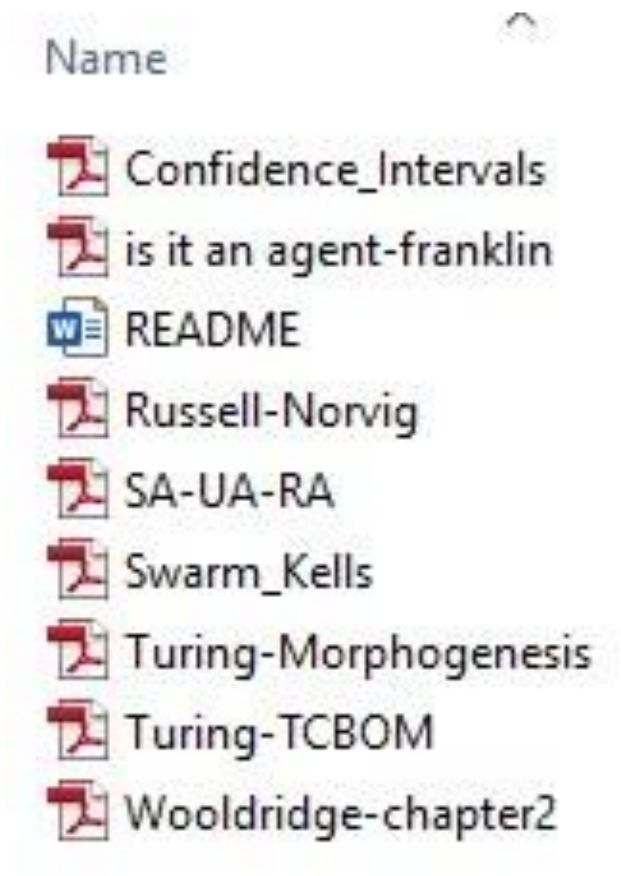
- An agent
- Autonomy
- Intelligence
- Communication
- Emergent Behavior
- Learning
- …

CS 6380: Logistics

http://www.cs.utah.edu/~tch/CS6380S2020/

CS 6380: Logistics

Also, look at Canvas files:



Name	
Confidence_Intervals	**Discussion of confidence intervals**
is it an agent-franklin	**Discussion of what an agent is**
README	
Russell-Norvig	**AI Text; lots on agents**
SA-UA-RA	**ABMS sensitivity analysis, etc.**
Swarm_Kells	**Discussion of flocking**
Turing-Morphogenesis	**Turing's reaction-diffusion paper**
Turing-TCBOM	
Wooldridge-chapter2	**Discussion of what an agent is**

CS 6380 Expectations

- You should have studied the assigned material before we cover it
- You must be able to develop Matlab implementations
- You are part of a learning team and should add value to the effort
- We may produce publishable results (through the project work)
- If you need help:
  - Ask questions in class
  - Send email
  - Arrange appointments

CS6380 Strategy

- Learning is organized around assignments
  - Start early
  - Iterate
  - You may send me solutions for review before the due date (this includes Lab Reports)
  - Specifications may be loose; part of the evaluation is how well you make decisions on how to proceed – assumptions, what to measure and how, statistical framework, etc.
  - One skill we aim to acquire is to convert mathematical or formal descriptions of processes into Matlab code that captures the semantics of the model

CS6380 Lab Reports

- Lab Reports
  - View lab report as a draft paper for submission for publication

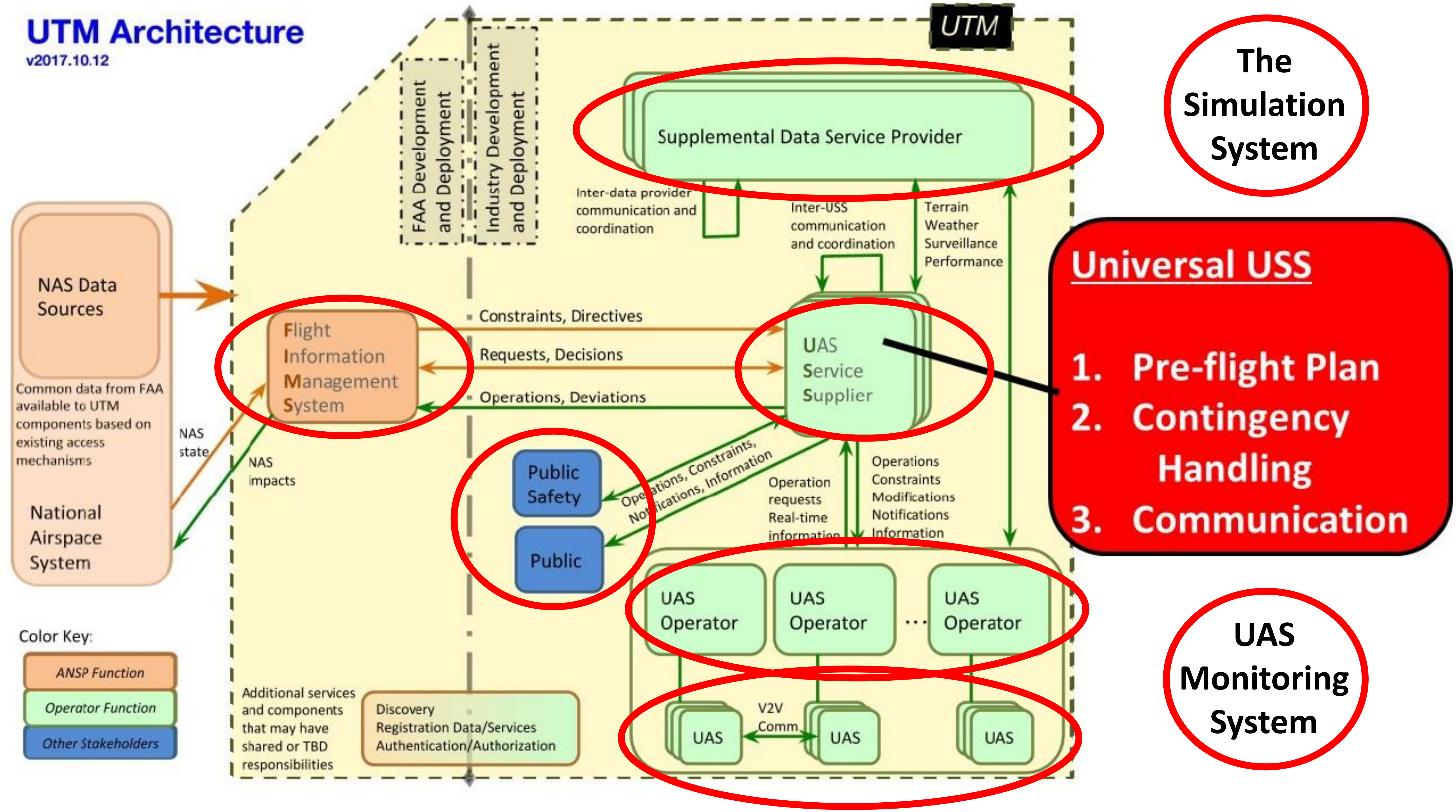
    http://www.cs.utah.edu/~tch/CS6380S2020/resources/lab-report-format
  - Important grading criteria given here:

    http://www.cs.utah.edu/~tch/CS6380S2020/resources/lab-criteria.pdf
  - Latex source here

    http://www.cs.utah.edu/~tch/CS6380S2020/resources/sample-lab-report/
  - You may submit them early for feedback

Project Domain: Urban Air Mobility

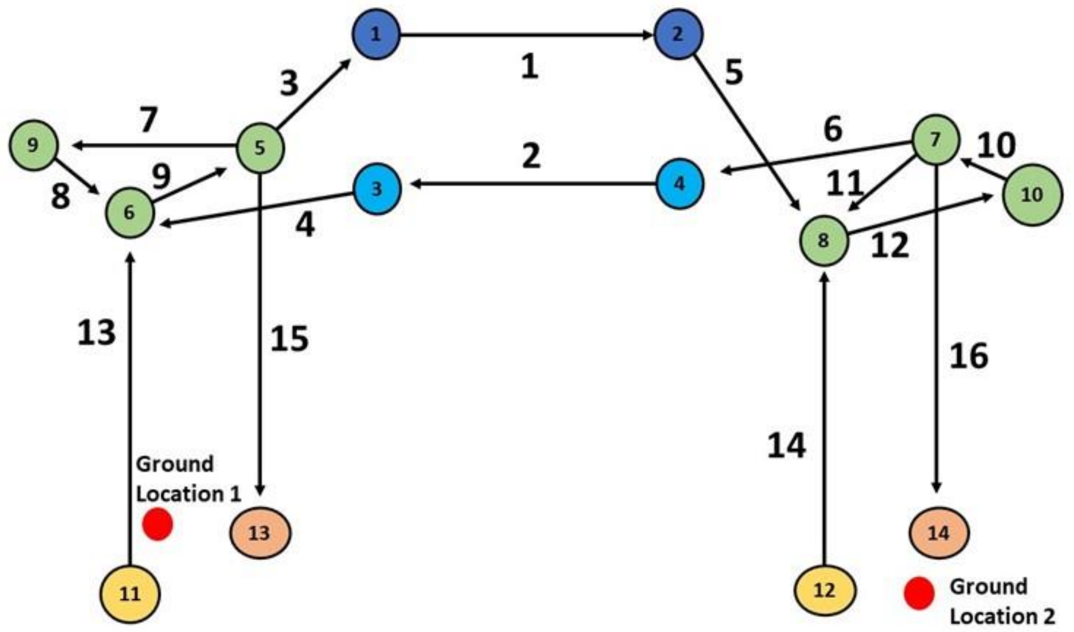
- The Utah Department of Transportation is looking to set up a way for thousands of Unmanned Aircraft Systems (UAS) to deliver packages, etc. over the Salt Lake Valley in the not too distant future.
- We are working on that with them.
- This involves a number of agents:
  - UAS
  - UAS Service Suppliers (USS)
  - Flight Reservation Systems
  - Flight Monitoring Systems
  - Supplemental Data Service Providers


UTM Architecture
v2017.10.12
UTM
FAA Development and Deployment
Industry Development and Deployment
Supplemental Data Service Provider
Inter-data provider communication and coordination
Inter-USS communication and coordination
Terrain
Weather
Surveillance
Performance
NAS Data Sources
Common data from FAA available to UTM components based on existing access mechanisms
National Airspace System
NAS state
NAS impacts
Flight Information Management System
Constraints, Directives
Requests, Decisions
Operations, Deviations
UAS Service Supplier
Operations, Constraints, Notifications, Information
Public Safety
Public
Operation requests
Real-time information
Operations
Constraints
Modifications
Notifications
Information
UAS Operator
UAS Operator
UAS Operator
UAS
V2V Comm
UAS
UAS
Color Key:
ANSP Function
Operator Function
Other Stakeholders
Additional services and components that may have shared or TBD responsibilities
Discovery
Registration Data/Services
Authentication/Authorization
The Simulation System
UAS Monitoring System

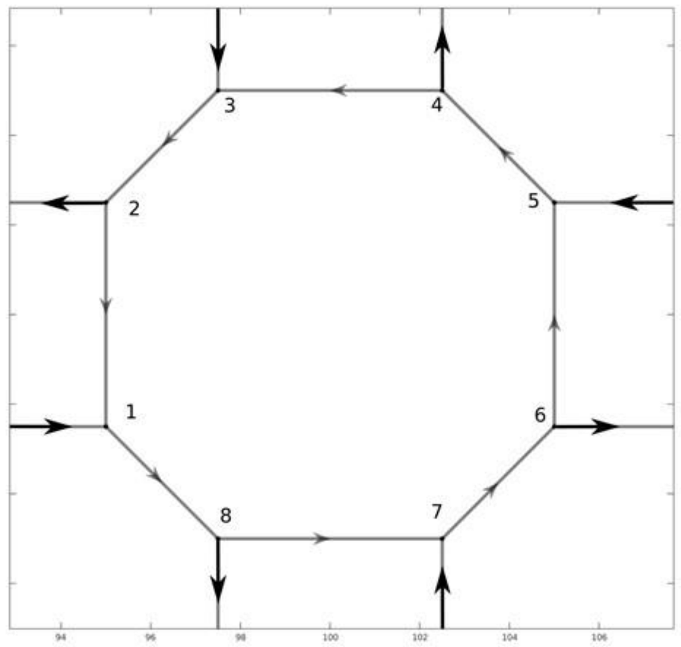
Universal USS

1. Pre-flight Plan
2. Contingency Handling
3. Communication

UAM: A Lane-Based Approach

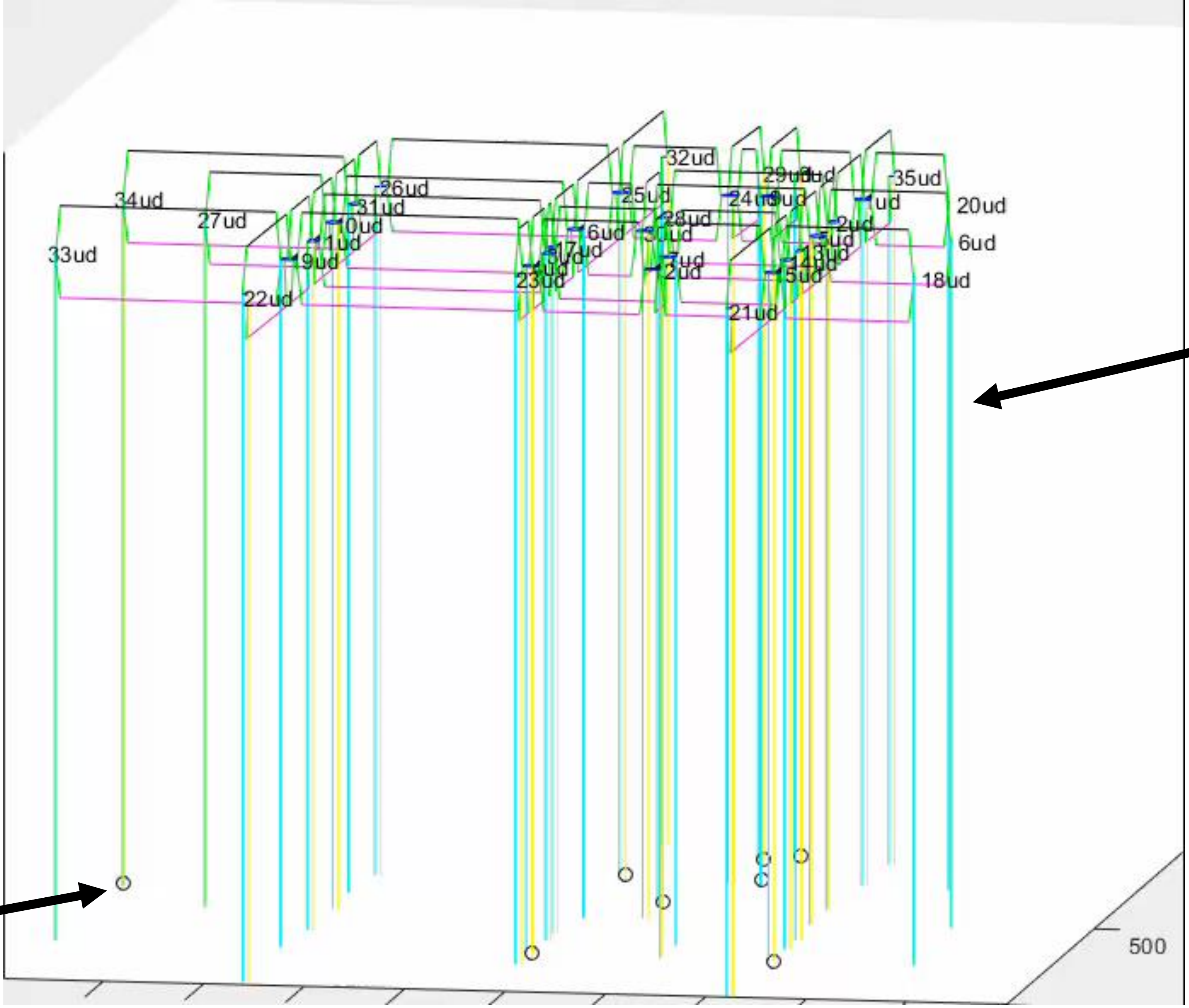


(a)



(b)

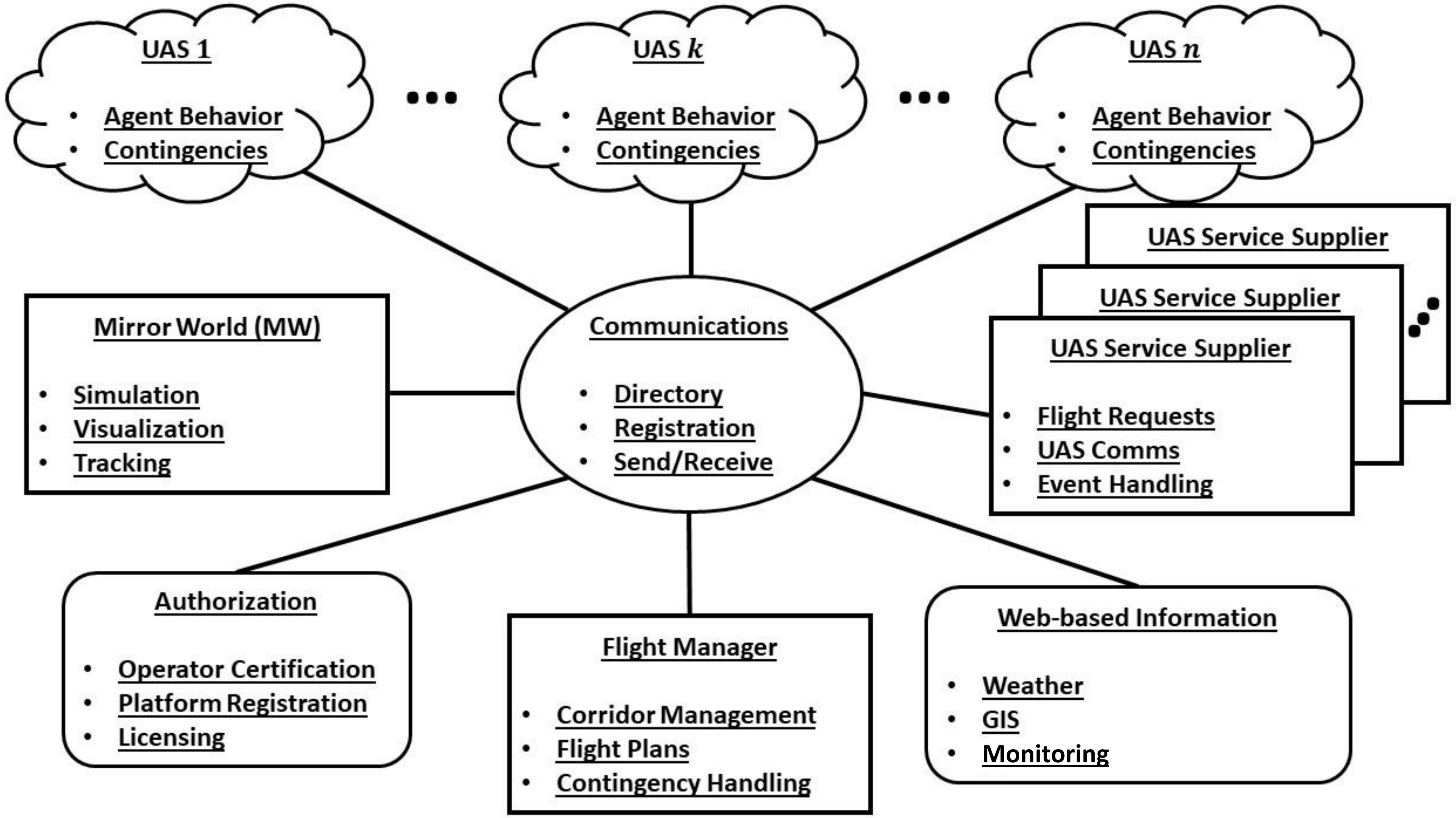
UAM System



Lanes

UAS

UTM Components (Agents)



Agents
(is-it-an-agent-Franklin.pdf)

Versions of agent characterization:

- Autonomous
- Domain oriented reasoning

- Perceives environment through sensors
- Acts on environment through effectors (actuators)

- Inhabit complex and dynamic world
- Autonomous
- Realizes set of goals or tasks

Agents
(is-it-an-agent-Franklin.pdf)

- Persistent

- Autonomy
- Social ability (communicates)
- Reactivity
- Pro-activeness

- Acts in real-world

Agents
(is-it-an-agent-Franklin.pdf)

Autonomous agent:

situated within and part of an environment

senses the environment

acts on the environment (over time)

pursues own agenda through actions

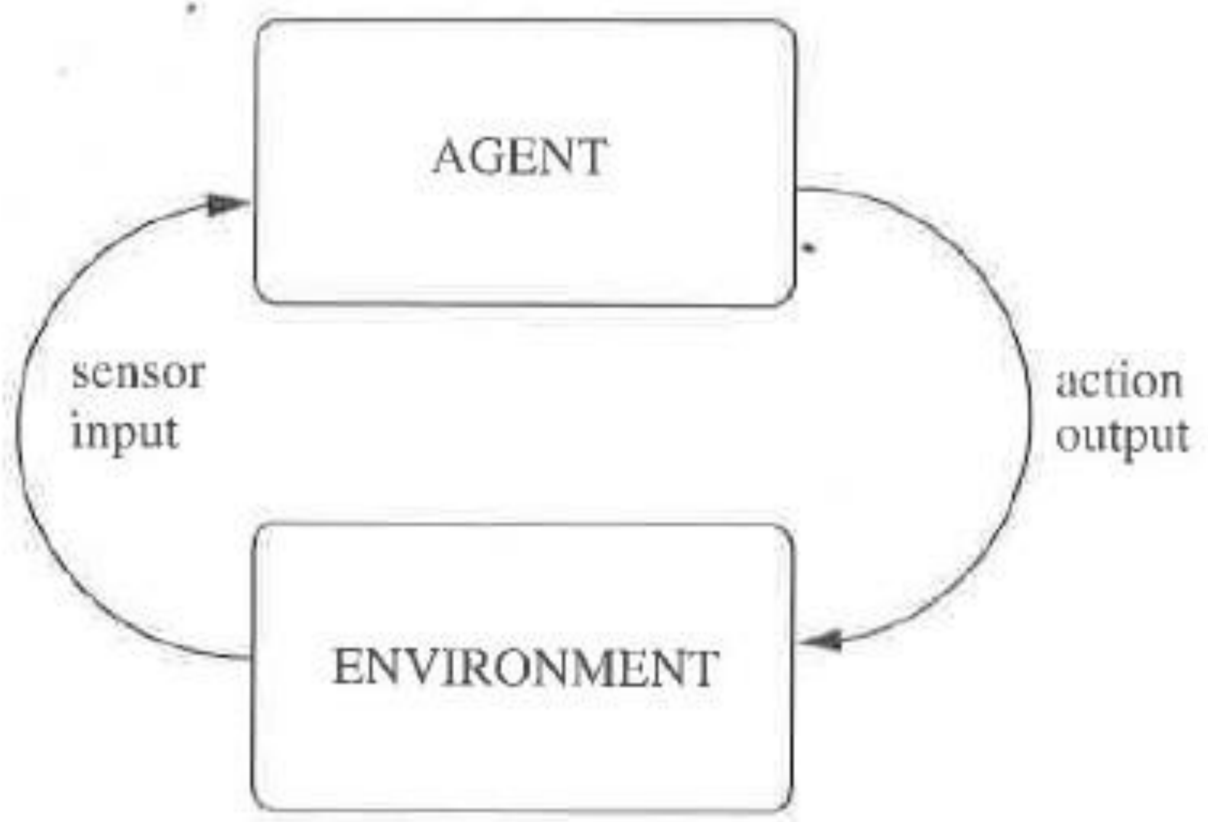
affects what it senses in the future.

Agents (Wooldridge-chapter2.pdf)

Agent:

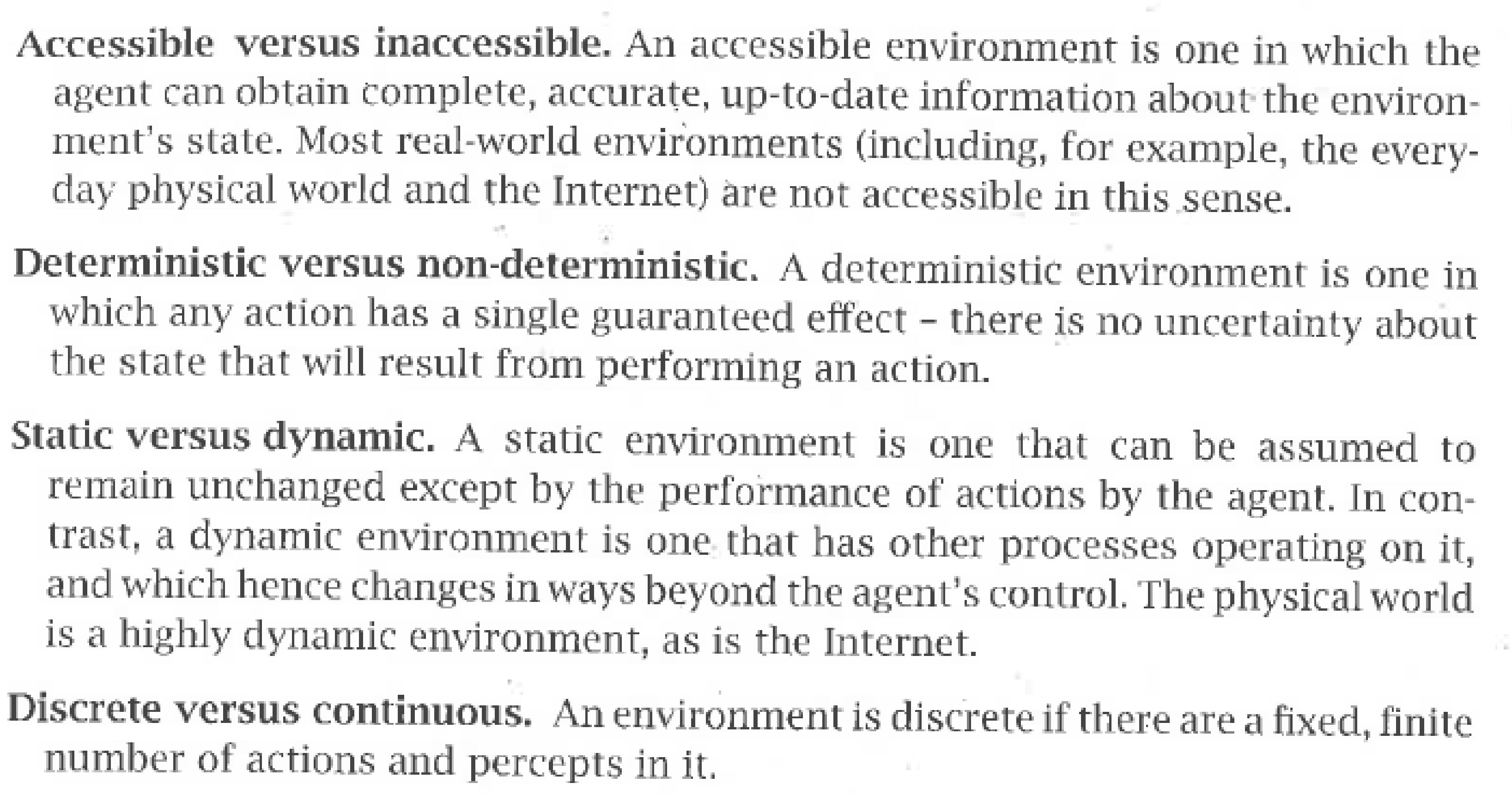
computer system
situated in some environment
capable of autonomous action in the environment
to meet its design objectives.

Agents (Wooldridge-chapter2.pdf)



Agents (Wooldridge-chapter2.pdf)

Agent Environment



Accessible versus inaccessible. An accessible environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state. Most real-world environments (including, for example, the everyday physical world and the Internet) are not accessible in this sense.

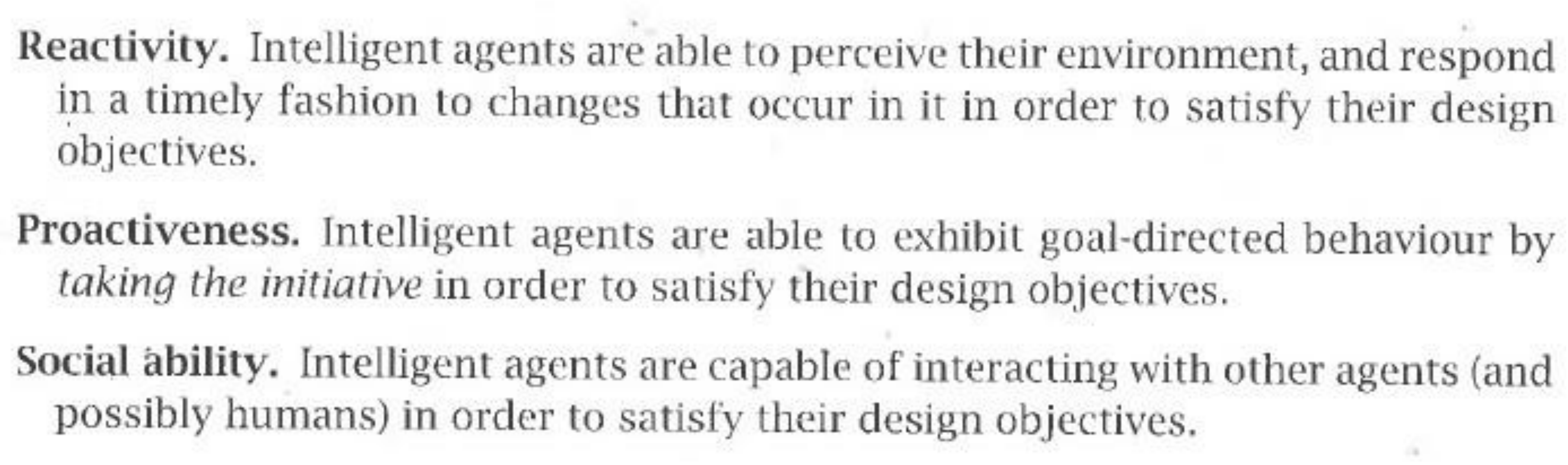
Deterministic versus non-deterministic. A deterministic environment is one in which any action has a single guaranteed effect – there is no uncertainty about the state that will result from performing an action.

Static versus dynamic. A static environment is one that can be assumed to remain unchanged except by the performance of actions by the agent. In contrast, a dynamic environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control. The physical world is a highly dynamic environment, as is the Internet.

Discrete versus continuous. An environment is discrete if there are a fixed, finite number of actions and percepts in it.

Agents (Wooldridge-chapter2.pdf)

Intelligent Agent



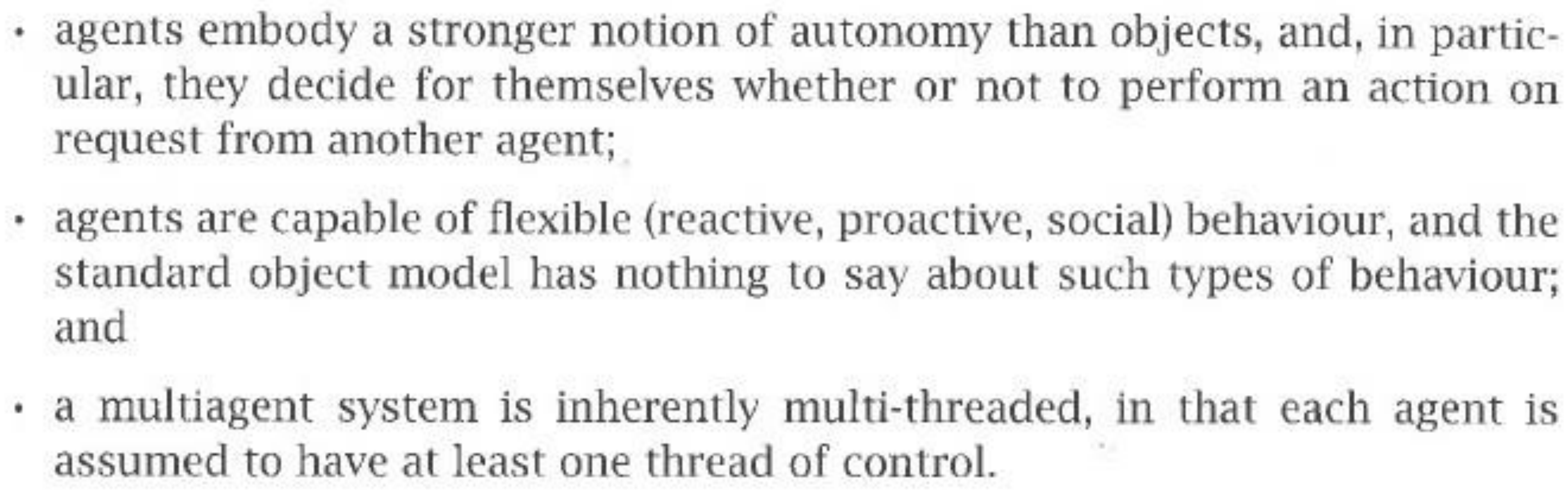
Reactivity. Intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives.

Proactiveness. Intelligent agents are able to exhibit goal-directed behaviour by *taking the initiative* in order to satisfy their design objectives.

Social ability. Intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

Agents (Wooldridge-chapter2.pdf)

Multiagent Systems



- agents embody a stronger notion of autonomy than objects, and, in particular, they decide for themselves whether or not to perform an action on request from another agent;
- agents are capable of flexible (reactive, proactive, social) behaviour, and the standard object model has nothing to say about such types of behaviour; and
- a multiagent system is inherently multi-threaded, in that each agent is assumed to have at least one thread of control.

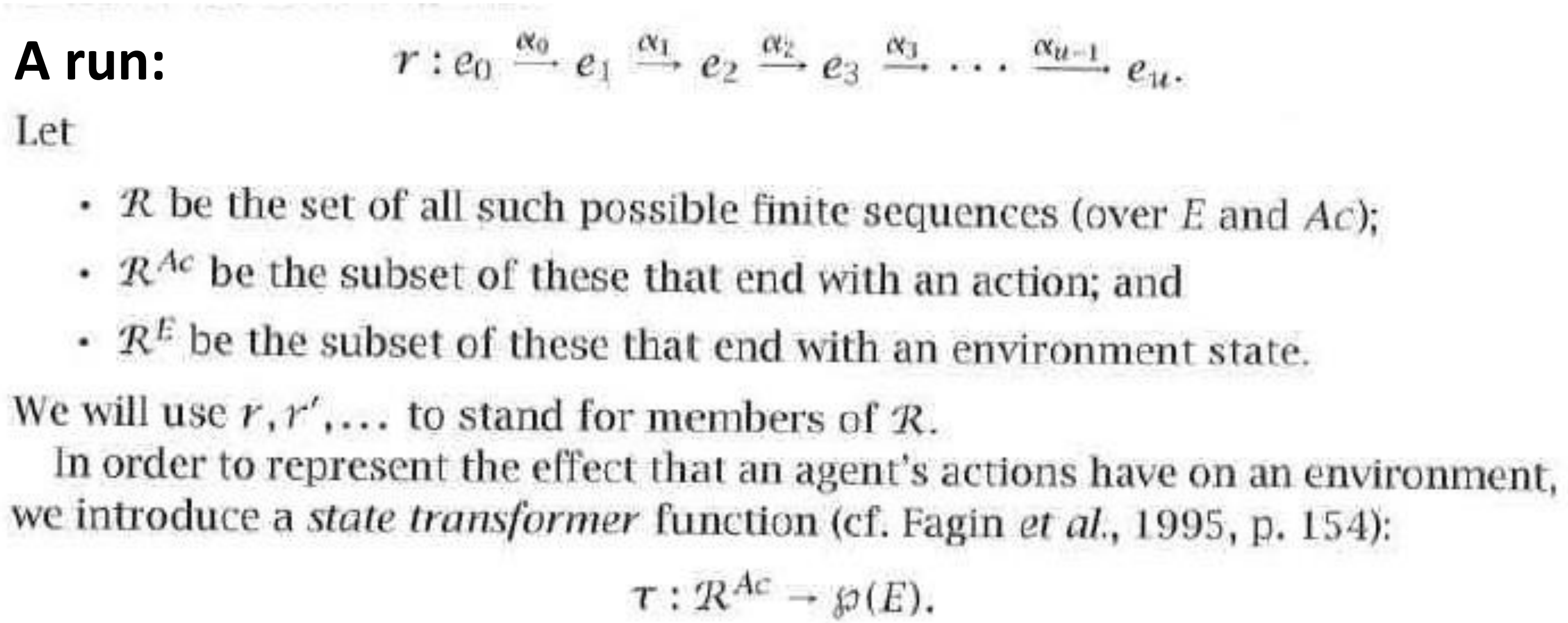
Agents (Wooldridge-chapter2.pdf)

Abstract Architectures for Agents

Environment States: $\boldsymbol{E = \{e, e', \ldots\}}$

Actions: **Ac** $= \{\boldsymbol{\alpha}, \boldsymbol{\alpha}', \ldots\}$

A run:



$$r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_{u-1}} e_u.$$

Let

- $\mathcal{R}$ be the set of all such possible finite sequences (over E and Ac);
- $\mathcal{R}^{Ac}$ be the subset of these that end with an action; and
- $\mathcal{R}^{E}$ be the subset of these that end with an environment state.

We will use $r, r', \ldots$ to stand for members of $\mathcal{R}$.

In order to represent the effect that an agent's actions have on an environment, we introduce a *state transformer* function (cf. Fagin *et al.*, 1995, p. 154):

$$\tau : \mathcal{R}^{Ac} \to \wp(E).$$

Agents (Wooldridge-chapter2.pdf)

Actions and Percepts for Agents

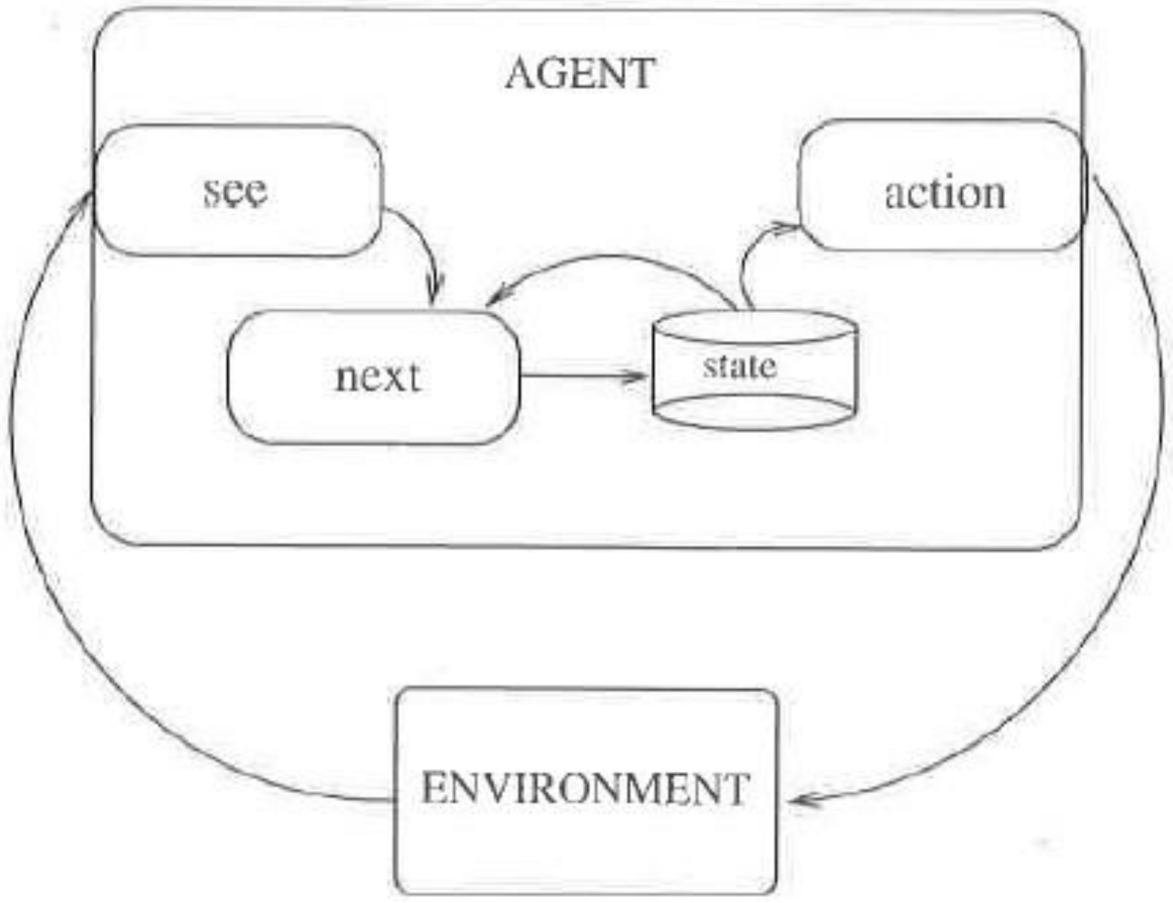
$$Ag : E \rightarrow Ac$$

Percepts function: $see : E \rightarrow Per$

Action function: $action : Per^{*} \rightarrow Ac$

Next state function: $next : I \times Per \rightarrow I$

Agents
(Wooldridge-chapter2.pdf)



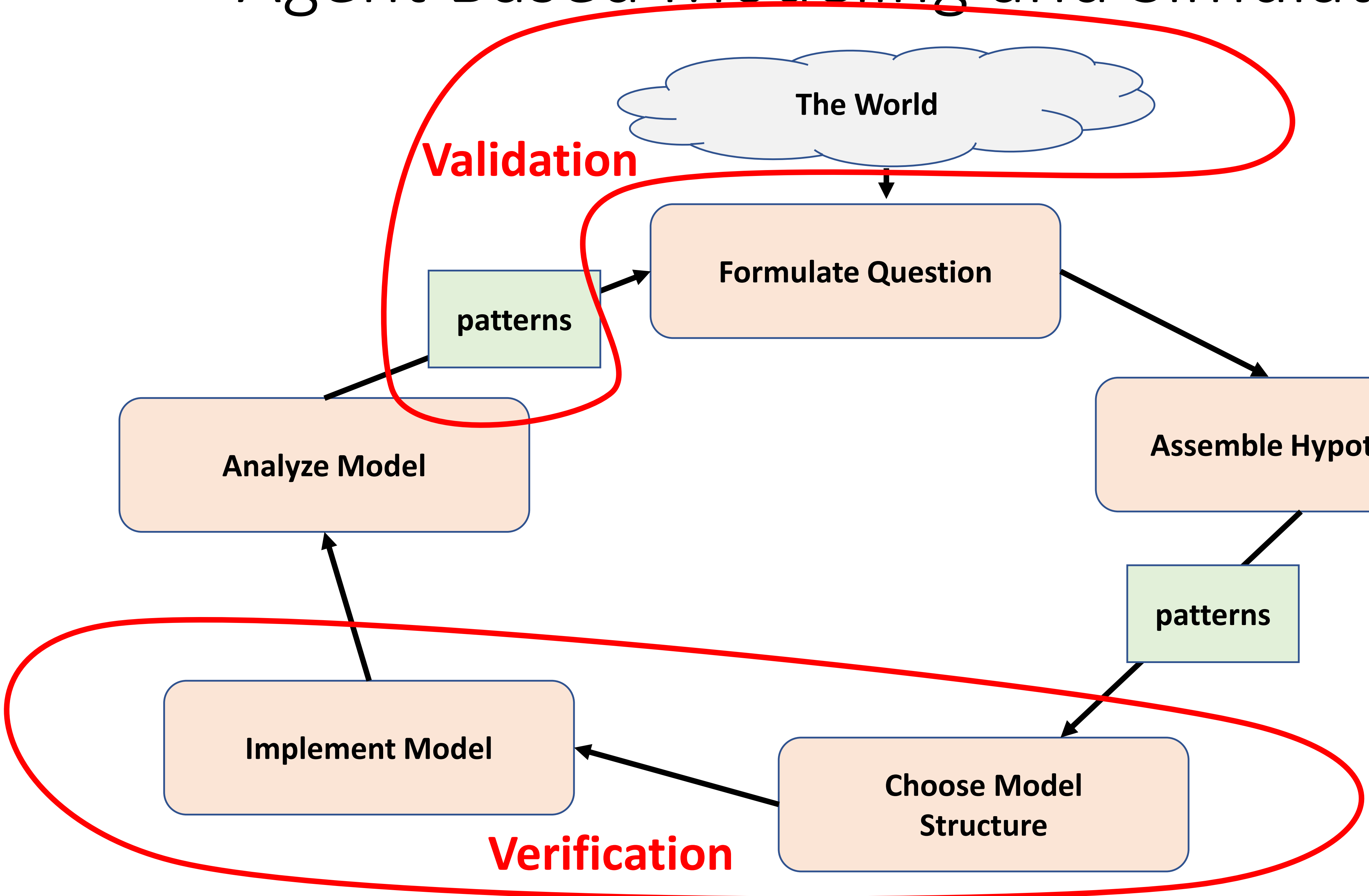
Agents: Matlab functions

```
function  action = CS6380_UAS1(percept)
%      → usually put full header
% if percept is 1, path is clear then go (return 1); else stop (return 0)

if percept==1
   action = 1;
else
   action = 0;
end
```

Agents: persistent state

```
function  action = CS6380_UAS1(percept)
%       → usually put full header
% always go forward first time into function
% if percept is 1, path is clear then go (return 1); else stop (return 0)

persistent first
action = 0;
if isempty(first)
   action = 1;
    first = 0;
    return
end
if percept==1
    action = 1;
end
```

Agent Based Modeling and Simulation



ABMS

- What local variables should be monitored?
- What global variables (patterns) should be monitored?
- How sensitive are global patterns to parameters?
- What role does uncertainty play in the input/output parameters?
- What structure is important to emergent pattern robustness?

➔*Sensitivity Analysis*: explores how small changes to input parameters impact emergent patterns

- ➔Needs full version of model
- ➔ reference parameter set
- ➔ one or two key outputs indicating behavior
- ➔ controlled simulation

➔*Uncertainty Analysis*: how uncertainty (variance) in output related to uncertainty in input

➔*Robustness Analysis*: How output patterns change with structural changes

Assignment A1: RD

http://www.cs.utah.edu/%7Etch/CS6380S2020/problems/A1/A1.pdf

Turing Reaction-Diffusion Patterns

diffusion

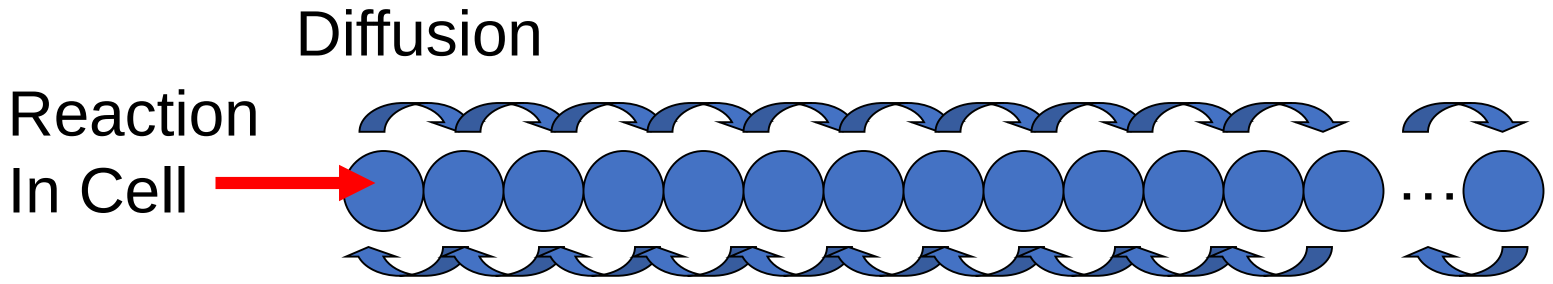
Formula:

$$\frac{\partial c}{\partial t} = \underbrace{f(c)}_{\text{reaction}} + \overbrace{D\nabla^2 c}^{\text{diffusion}}$$

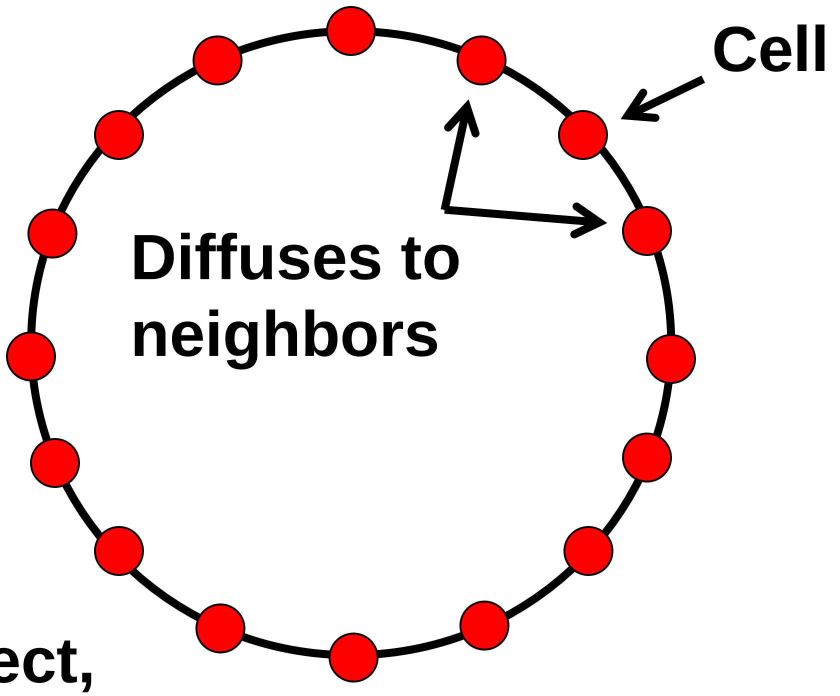
reaction

Simple Scenario (from Turing’s 1952 paper)

A set of neighboring cells, each cell runs reaction and diffuses to neighbors



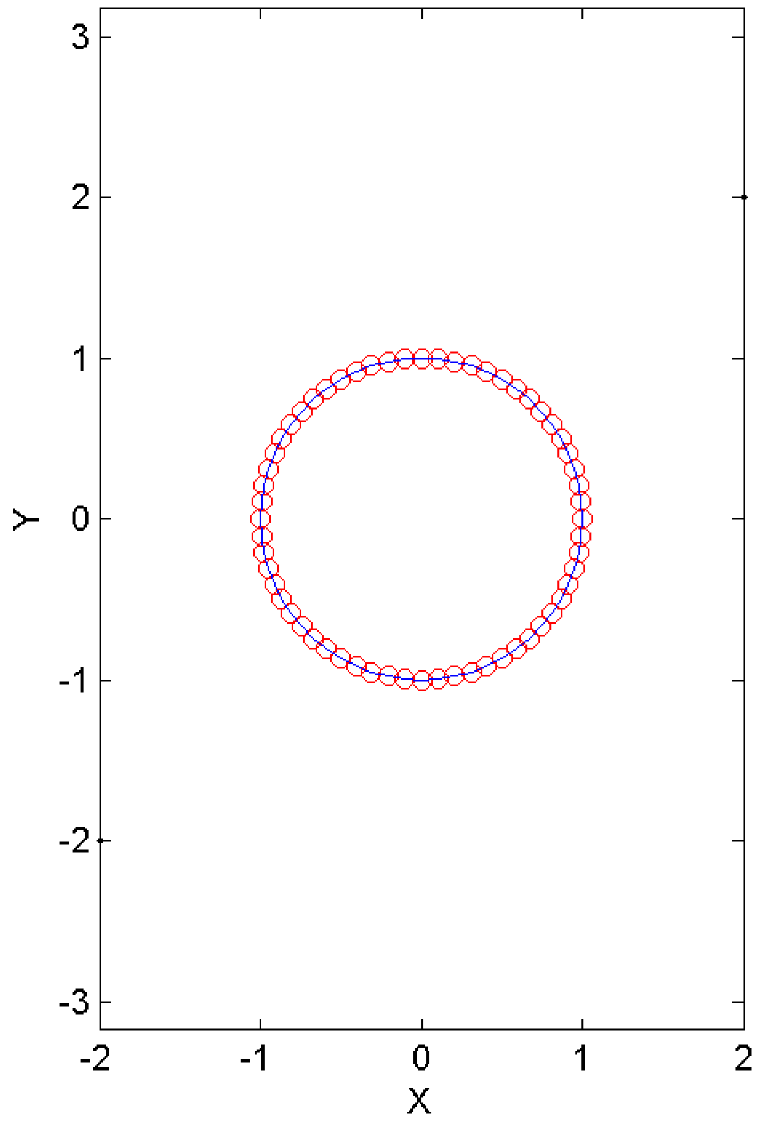
Consider Circular Layout

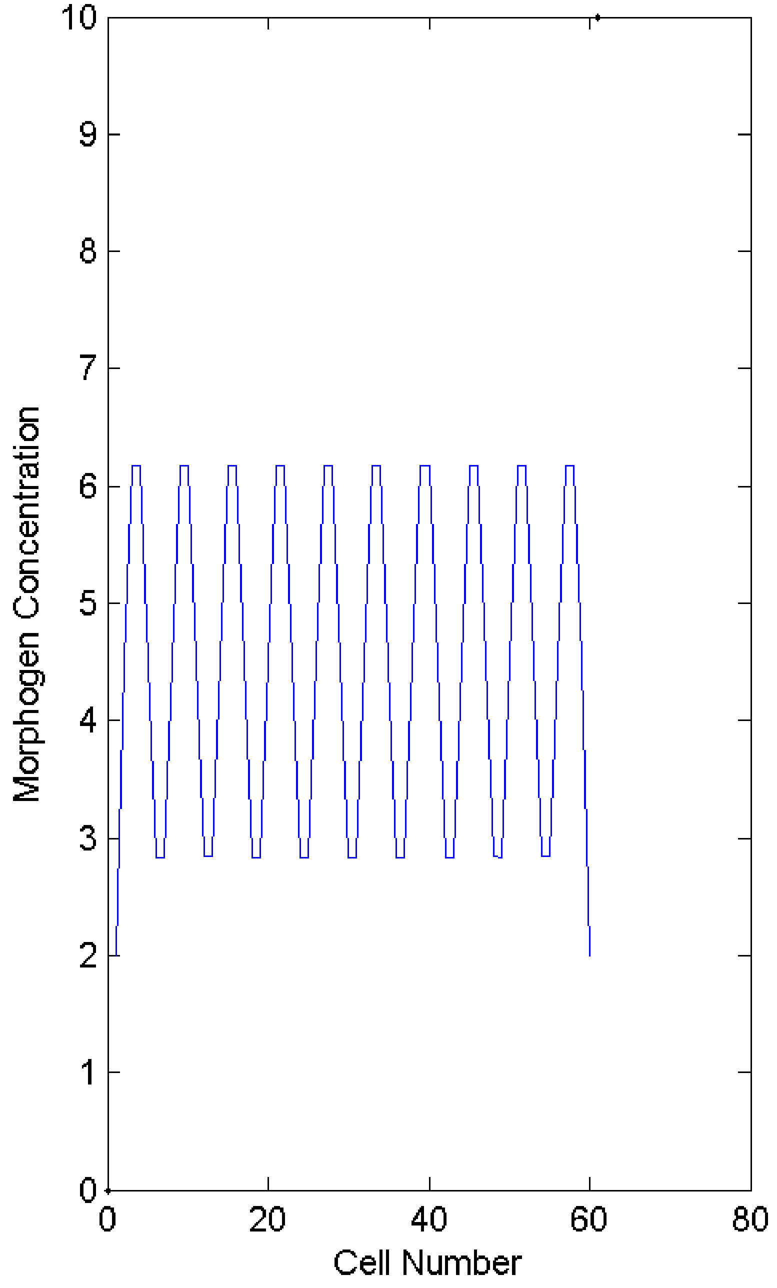


If layout is regular and topology correct, then pattern forms as desired

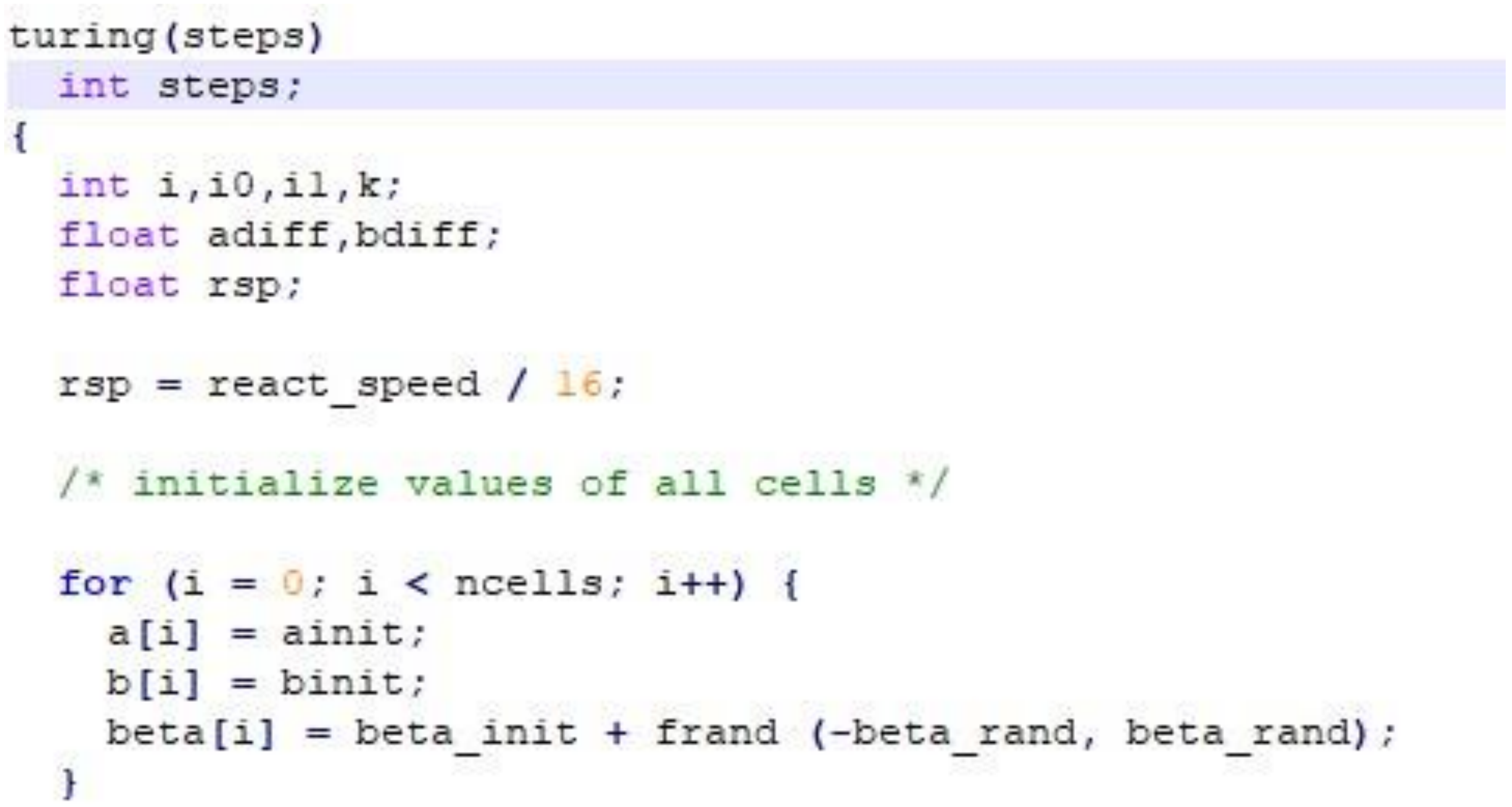
If not …

No error in placement



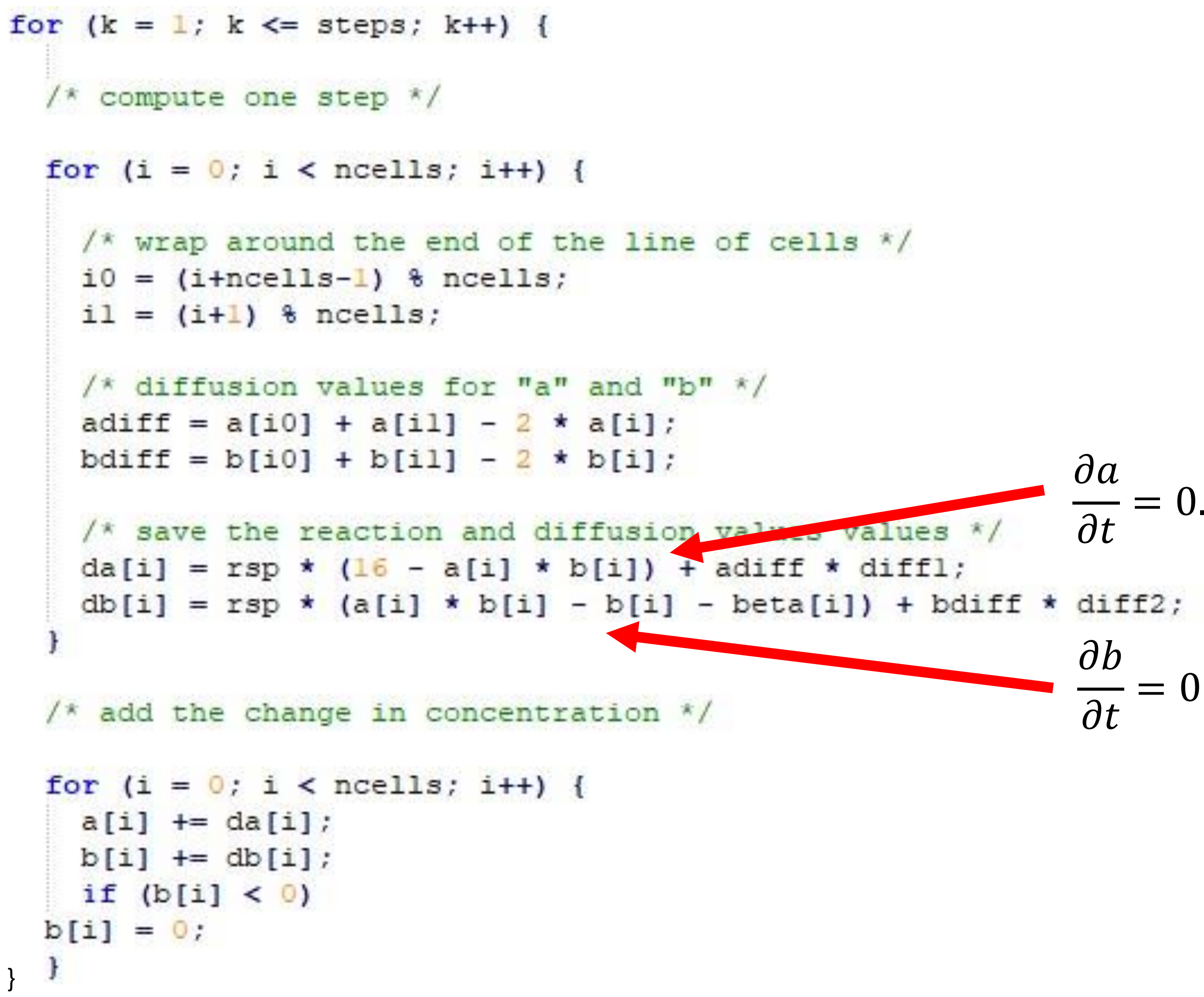


Reaction-Diffusion Code: Initialize



```
turing(steps)
  int steps;
{
  int i,i0,i1,k;
  float adiff,bdiff;
  float rsp;

  rsp = react_speed / 16;

  /* initialize values of all cells */

  for (i = 0; i < ncells; i++) {
    a[i] = ainit;
    b[i] = binit;
    beta[i] = beta_init + frand (-beta_rand, beta_rand);
  }
```

Reaction-Diffusion Code: Run Process

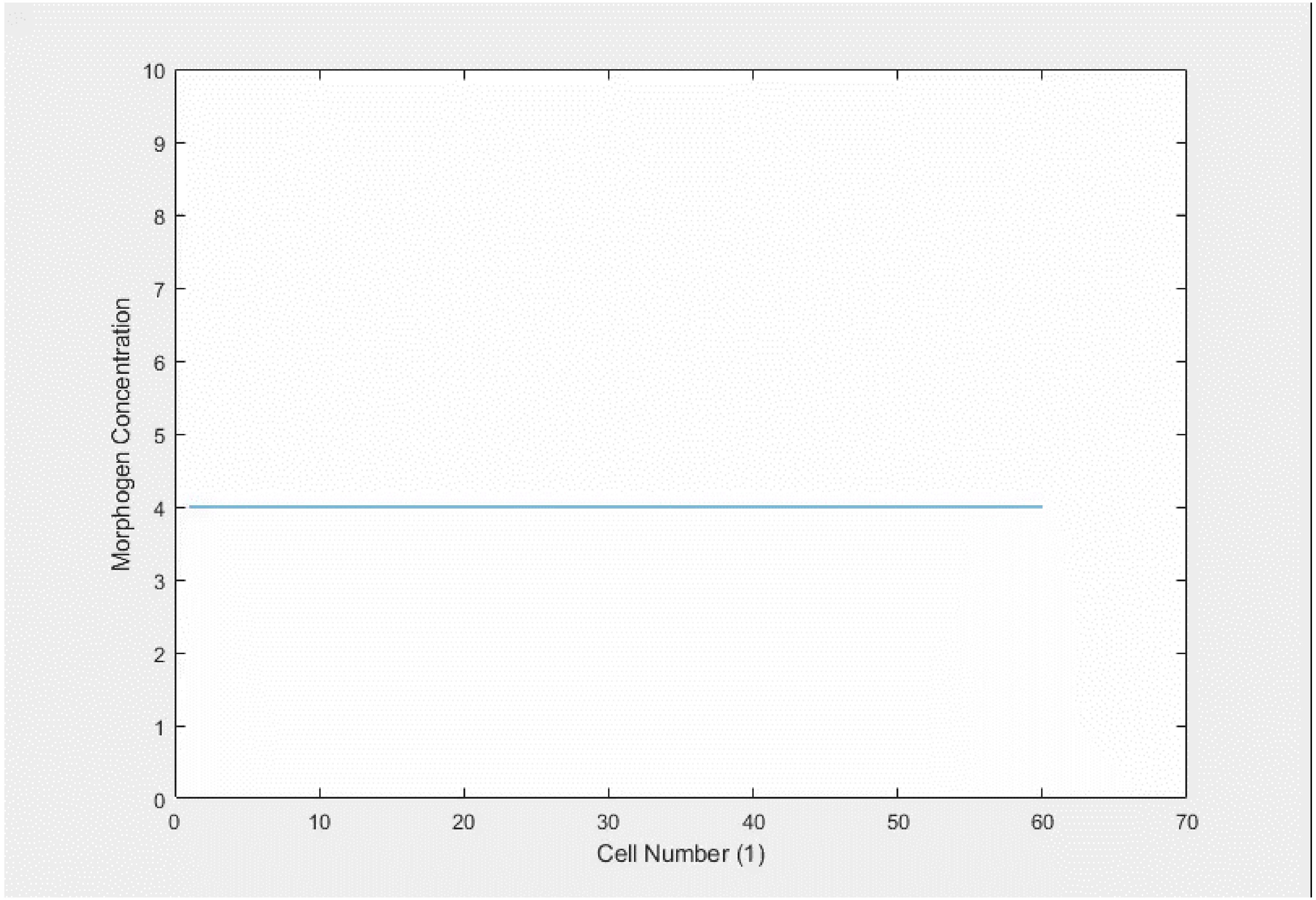


```
for (k = 1; k <= steps; k++) {

  /* compute one step */

  for (i = 0; i < ncells; i++) {

    /* wrap around the end of the line of cells */
    i0 = (i+ncells-1) % ncells;
    i1 = (i+1) % ncells;

    /* diffusion values for "a" and "b" */
    adiff = a[i0] + a[i1] - 2 * a[i];
    bdiff = b[i0] + b[i1] - 2 * b[i];

    /* save the reaction and diffusion values values */
    da[i] = rsp * (16 - a[i] * b[i]) + adiff * diff1;
    db[i] = rsp * (a[i] * b[i] - b[i] - beta[i]) + bdiff * diff2;
  }

  /* add the change in concentration */

  for (i = 0; i < ncells; i++) {
    a[i] += da[i];
    b[i] += db[i];
    if (b[i] < 0)
  b[i] = 0;
  }
}
```

$$\frac{\partial a}{\partial t} = 0.0625(16 - a_i * b_i) + 0.25(a_{i-1} + a_{i+1} - 2a_i)$$

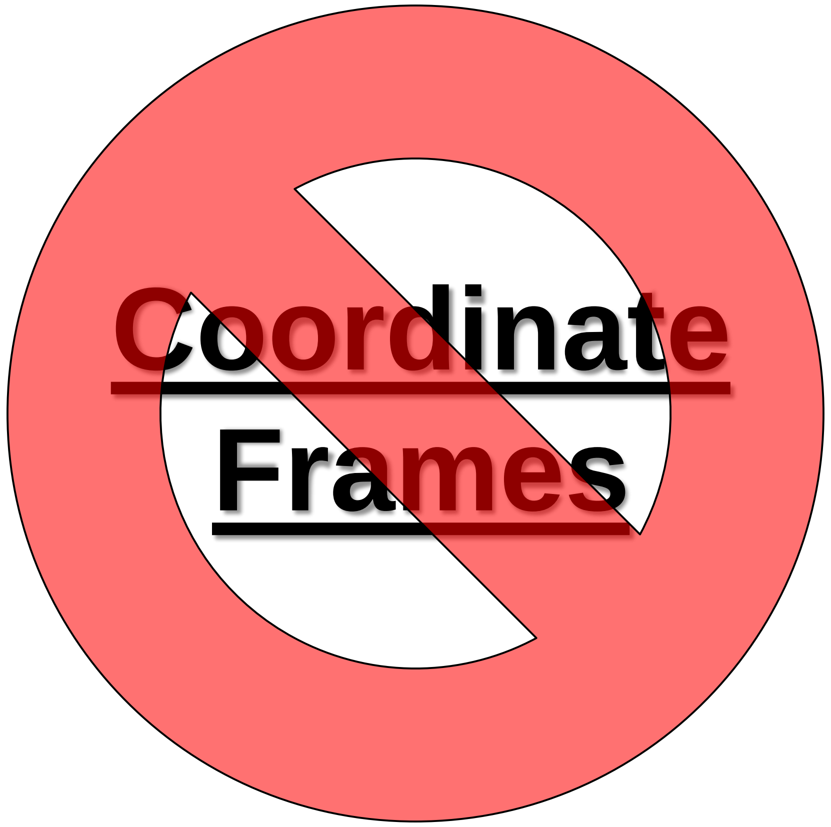
$$\frac{\partial b}{\partial t} = 0.0625(a_i * b_i - b_i - \beta_i) + 0.0625(b_{i-1} + b_{i+1} - 2b_i)$$

RD Process (60 cells)



Topology vs. Metrology

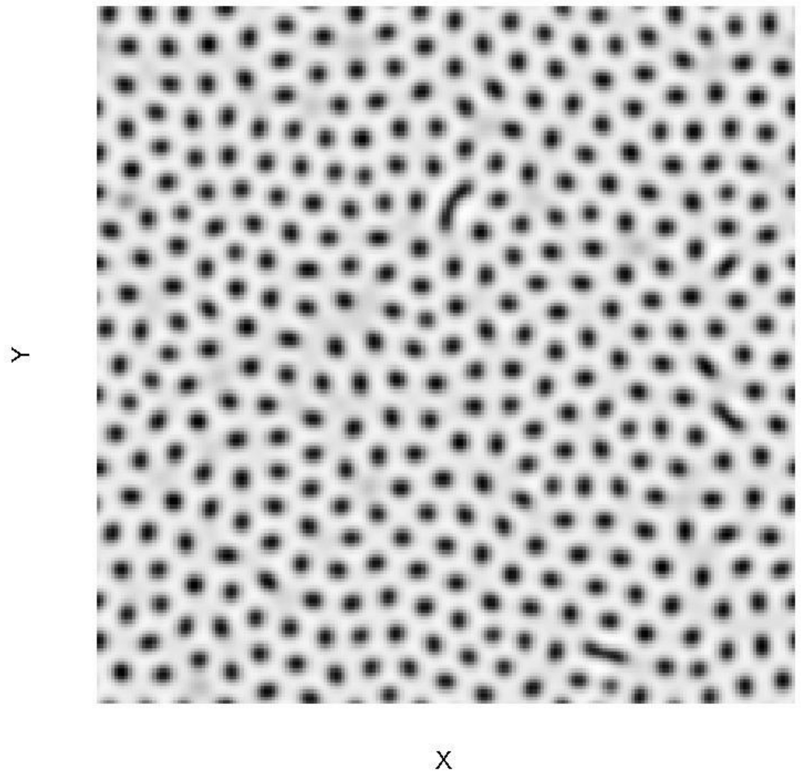
- RD patterns form:
  - at the cell level
  - NOT wrt a fixed coordinate frame
- Do not compute the actual Laplacian: there's no continuous function

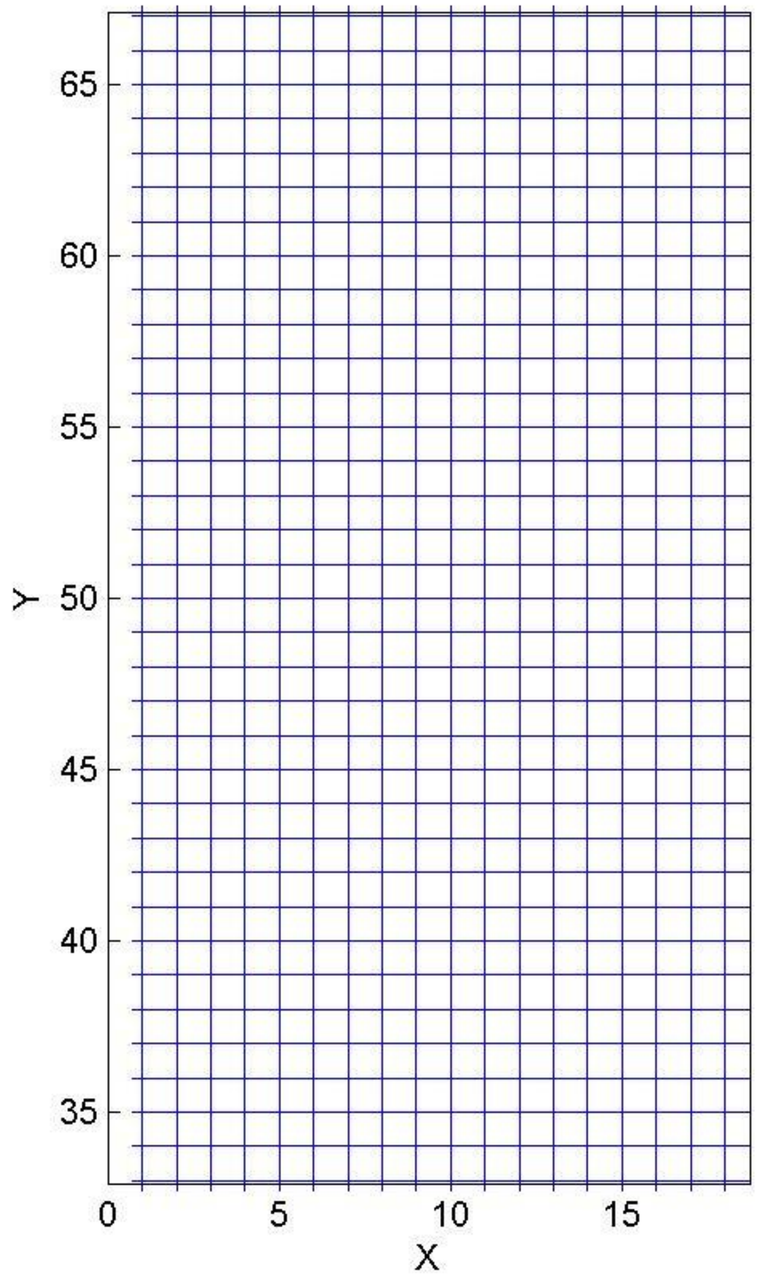


Topological Issues

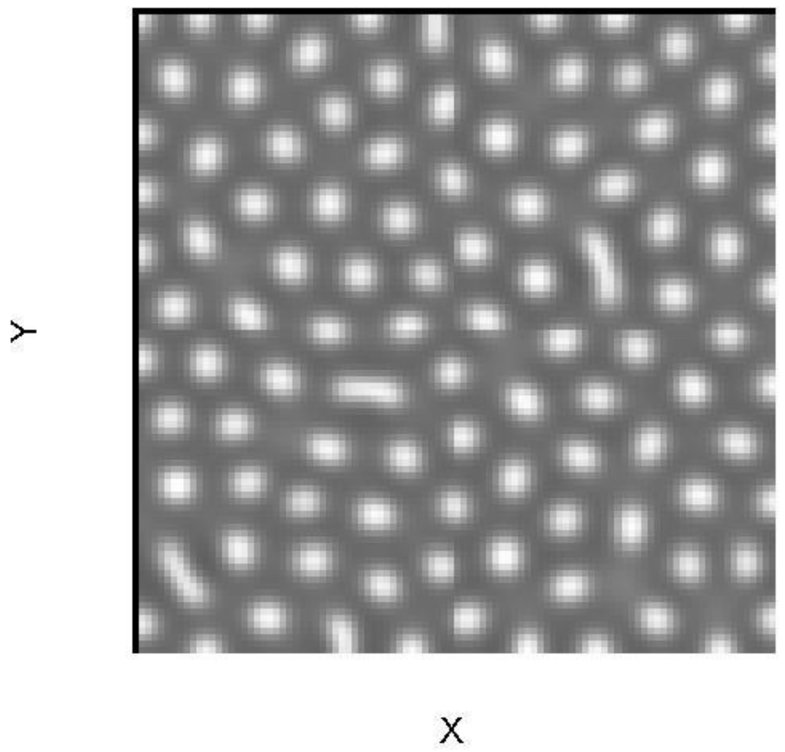
- Network connectivity impacted by communication properties
  - RD computation based on neighbors
  - Do not use location or distance
- Pattern impacted by neighborhood relations
  - Will pattern form? (Frequency?)
  - Rate of convergence?
  - Etc.

Turing R-D for 2D Spots

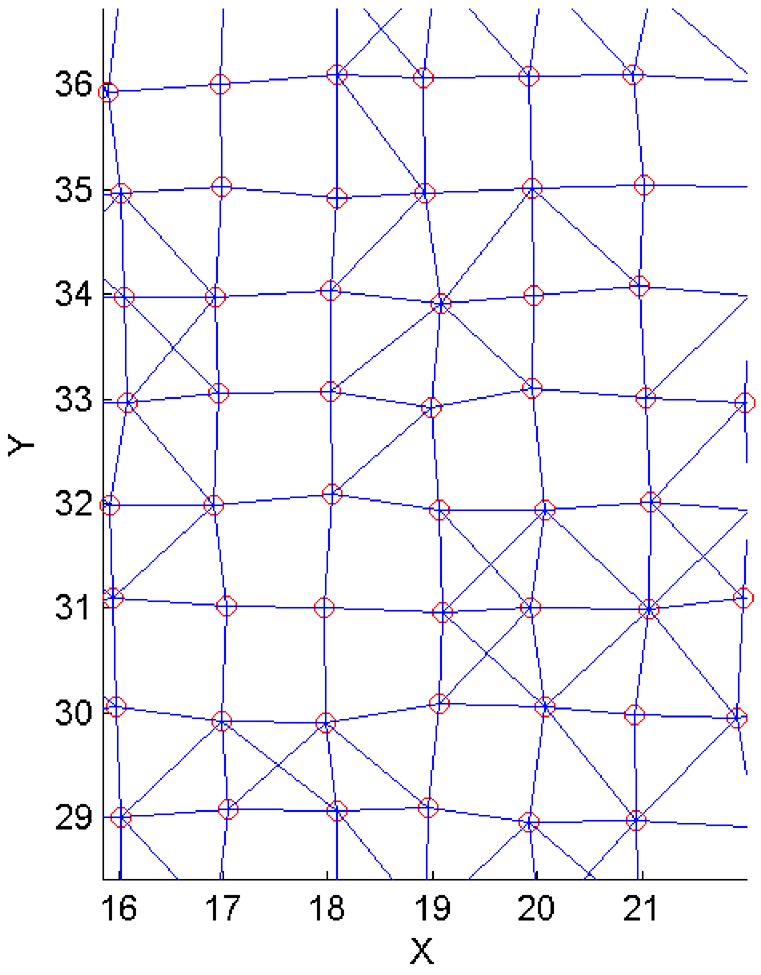


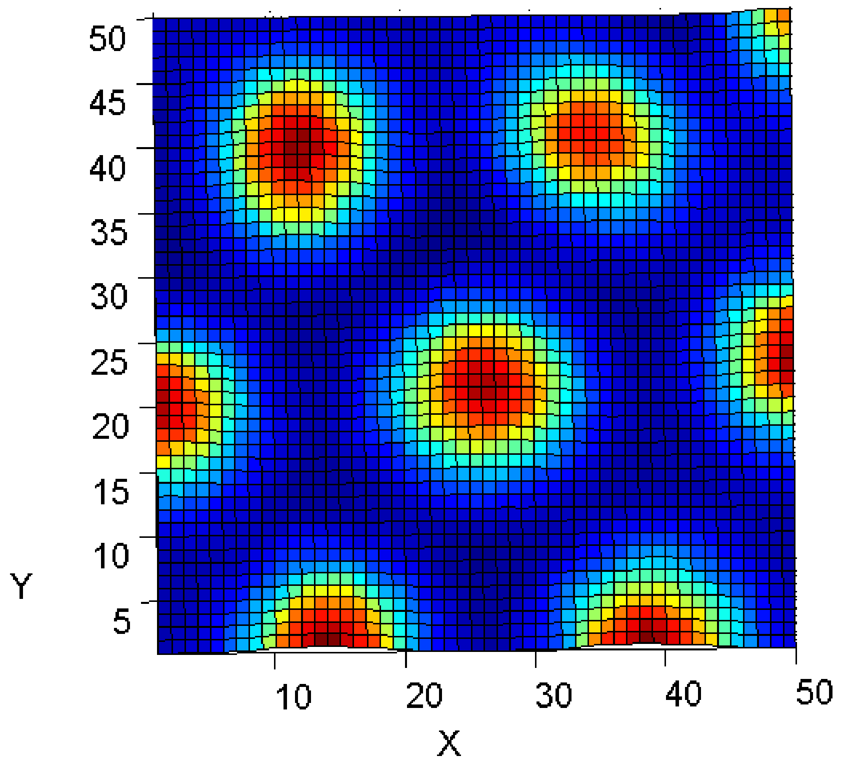


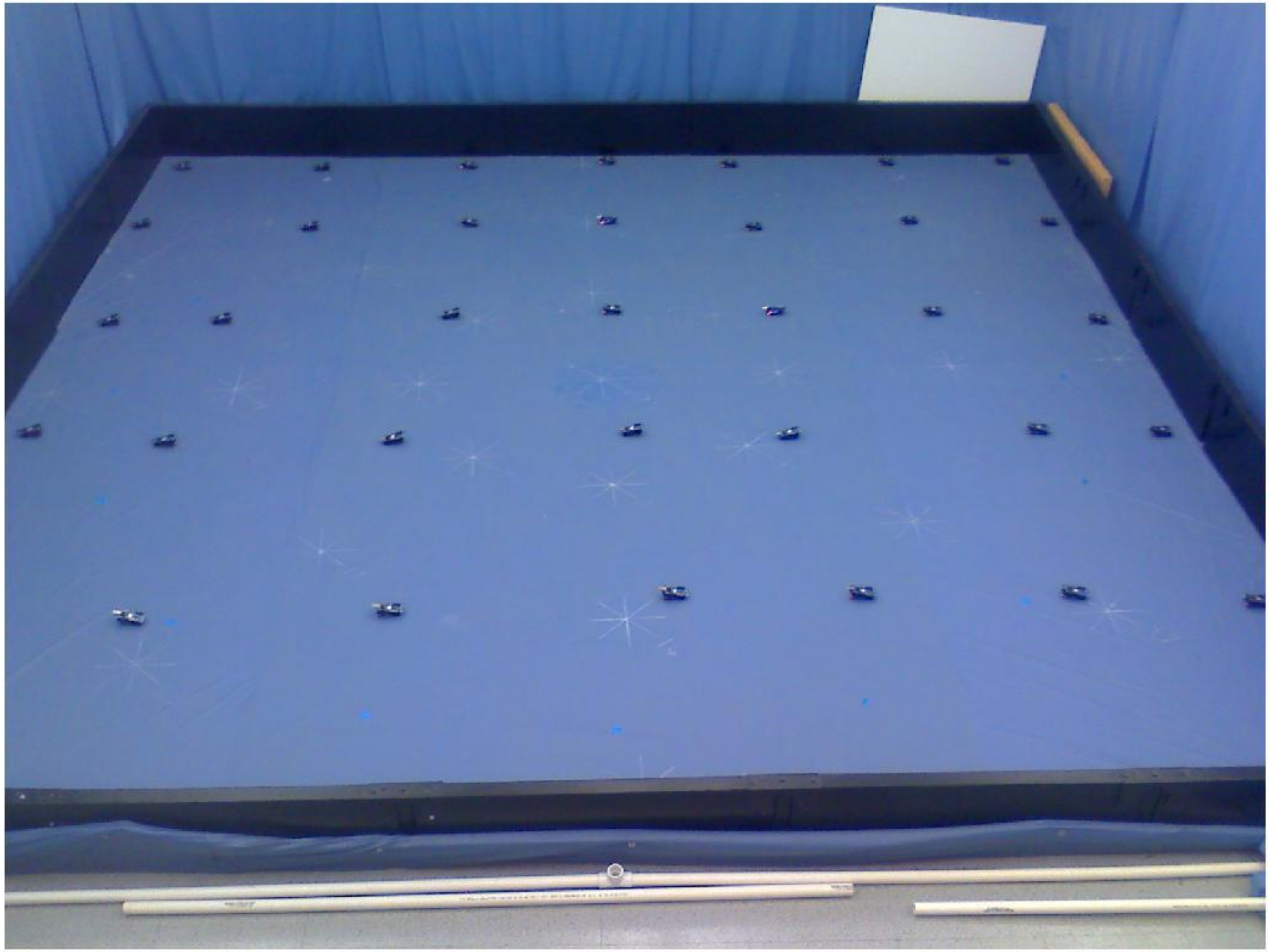
65
60
55
50
45
40
35
Y
0
5
10
15
X



Y
X

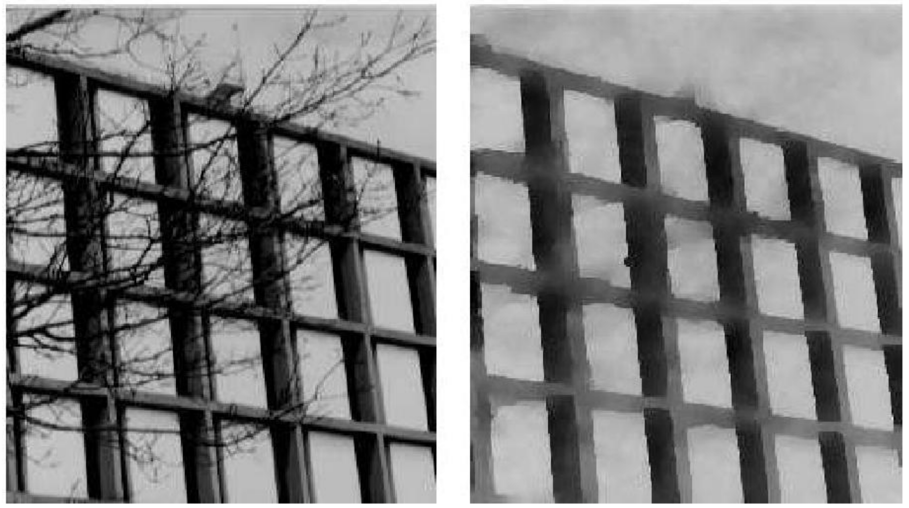

36
35
34
33
32
31
30
29
Y
16
17
18
19
20
21
X


50
45
40
35
30
25
20
15
10
5
Y
10
20
30
40
50
X



RD Sensor MoteTesting Layout

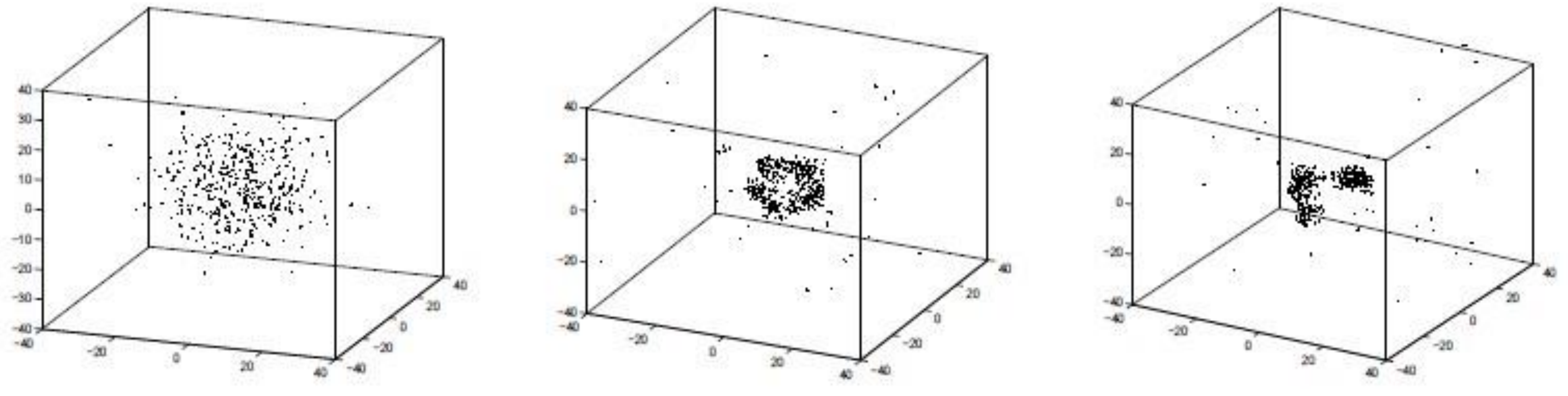
Zhu and Mumford R-D Camouflage



Tree branches
are removed by
simulating texture
of background using
reaction-diffusion
process.

Flocking

1. *Alignment:* Elements move towards the average heading of local flock-mates.
2. *Cohesion:* Elements move towards the average position of local flock-mates
3. *Separation:* Elements of the swarm don't collide with each other. (Unless they're stupid — this is easier to program than you might think).



(a) The initial position of a swarm (b) After 50 time steps (c) After 100 time steps

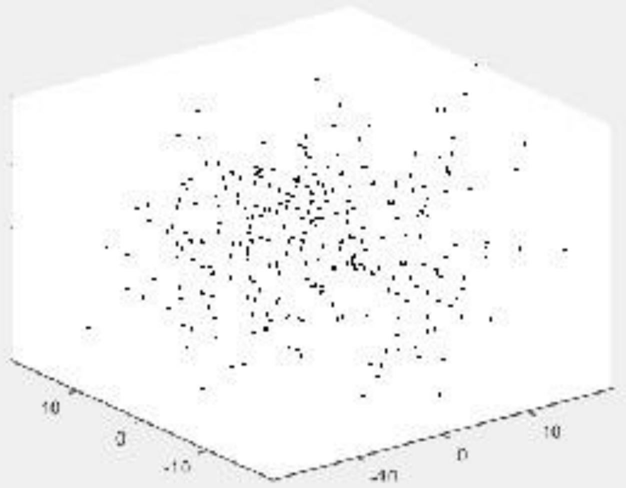
Figure 1: A swarm for N=400, c1=0.3, c2=0.3,c3=5,c4=5; A=10, V=10 and only cares about objects within a radius of 10 to the front and side.

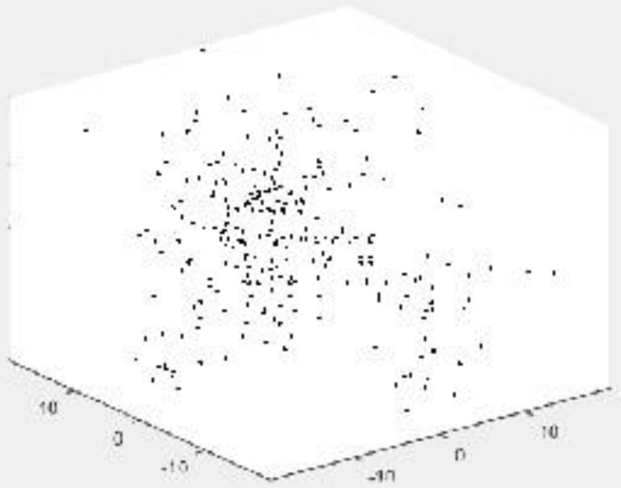
Flocking

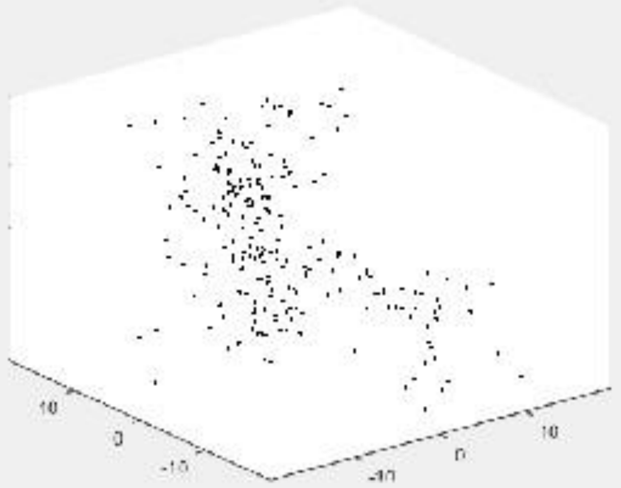
Issues:
- **delta (0.05)**
- **number of neighbors (3)**

Otherwise:

different results!

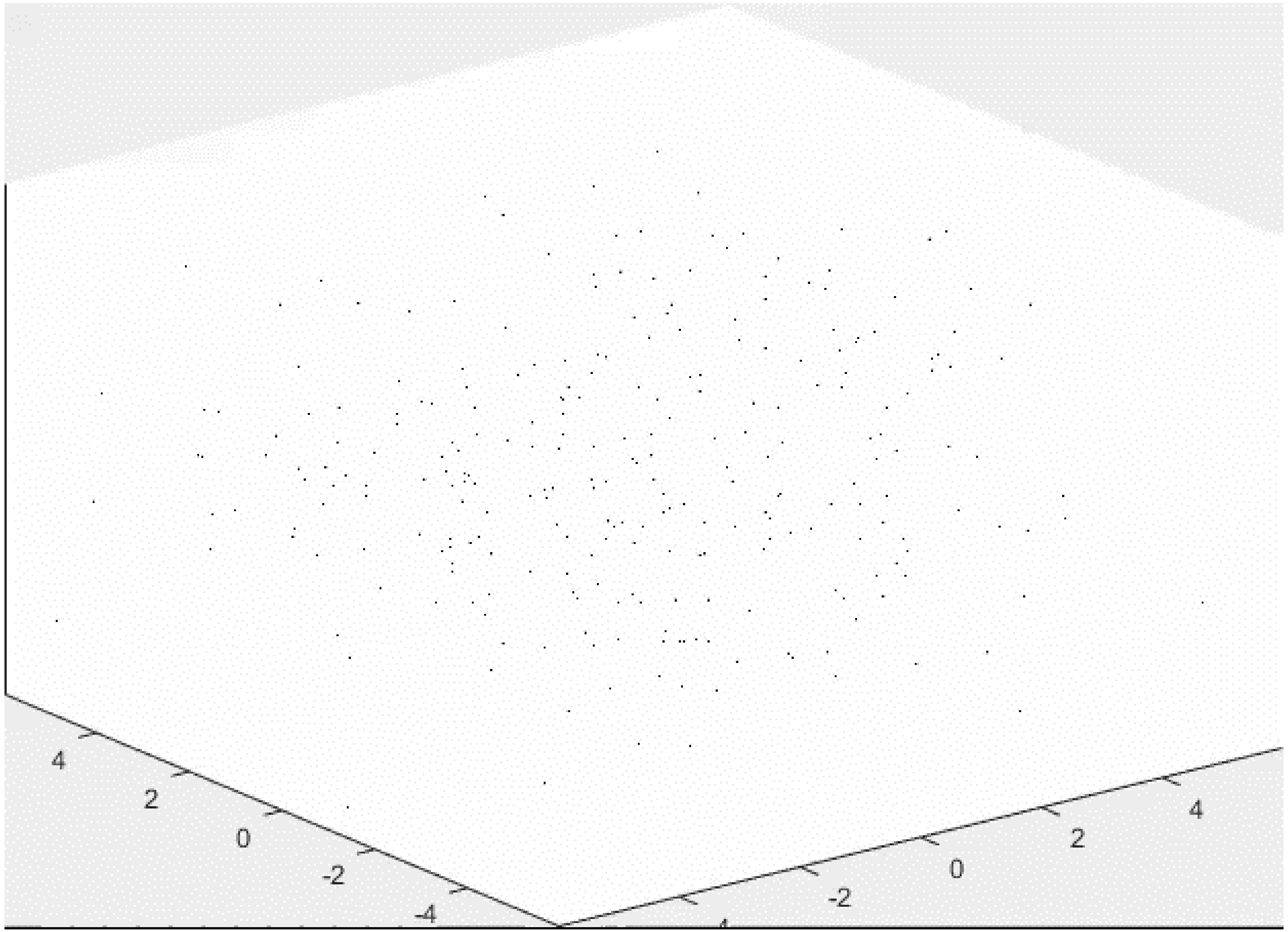






Initial State **Step 50** **Step 100**

Flocking Example



Agents: Russell & Norvig

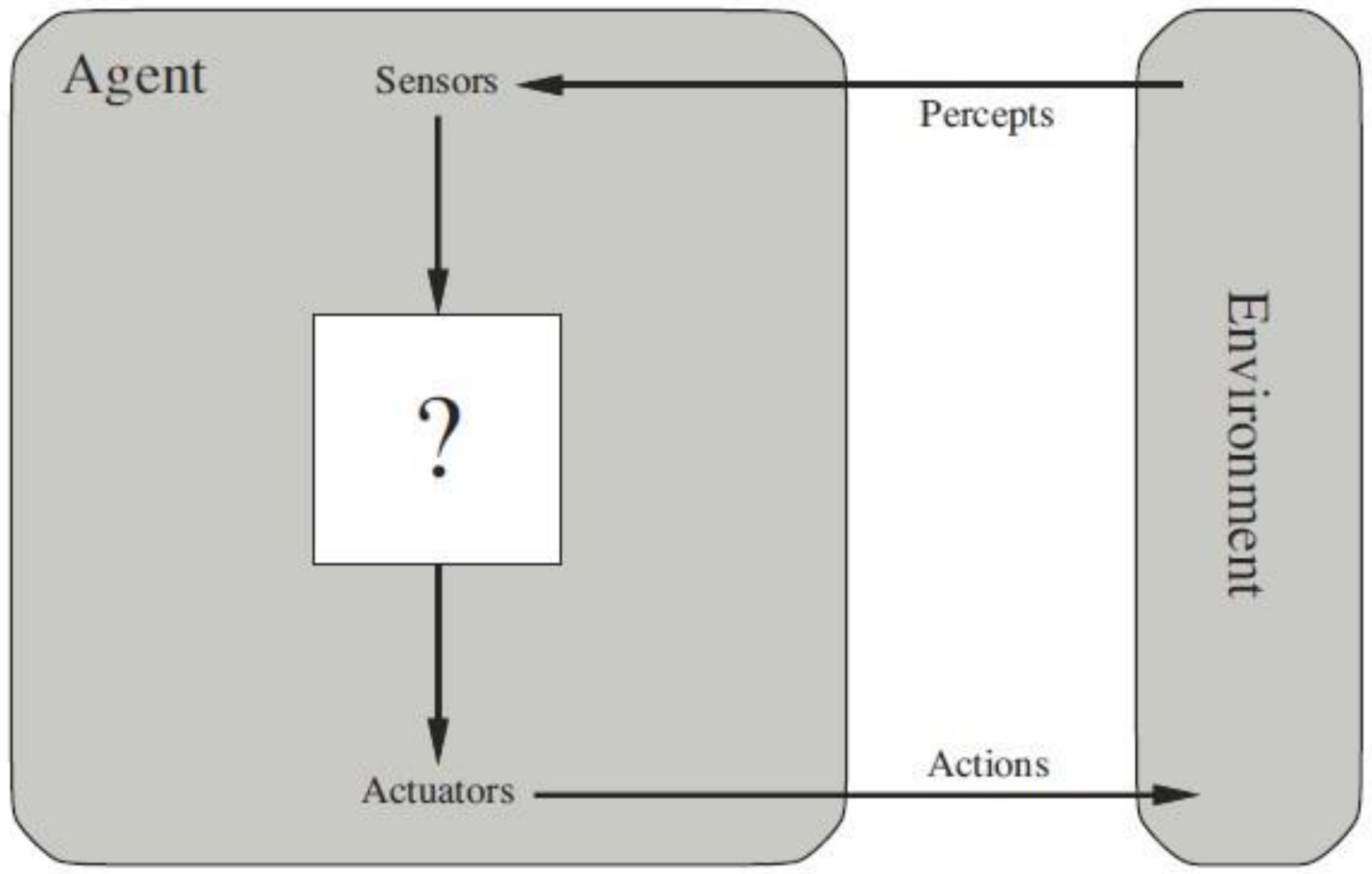
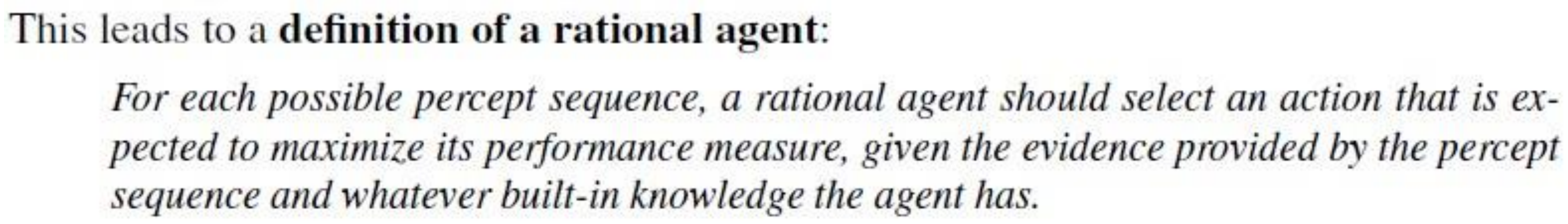


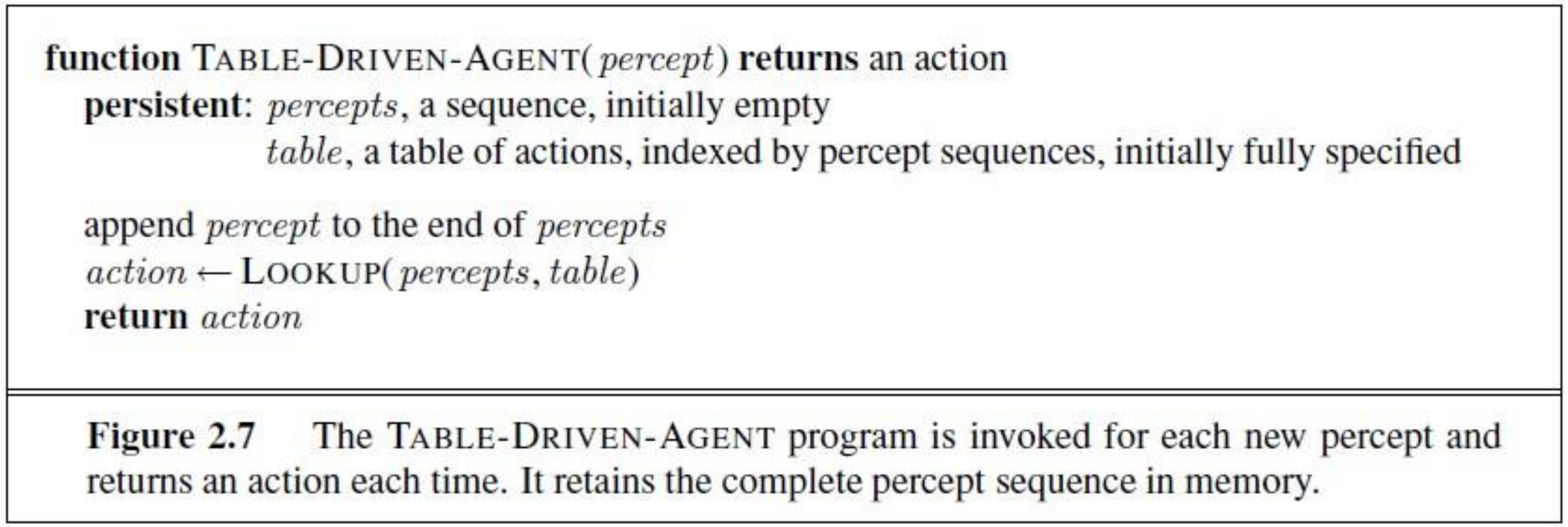
Figure 2.1 Agents interact with environments through sensors and actuators.

Agents: Russell & Norvig



This leads to a **definition of a rational agent**:

> *For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.*



```
function TABLE-DRIVEN-AGENT(percept) returns an action
  persistent: percepts, a sequence, initially empty
              table, a table of actions, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

Figure 2.7 The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

Agents: Russell & Norvig

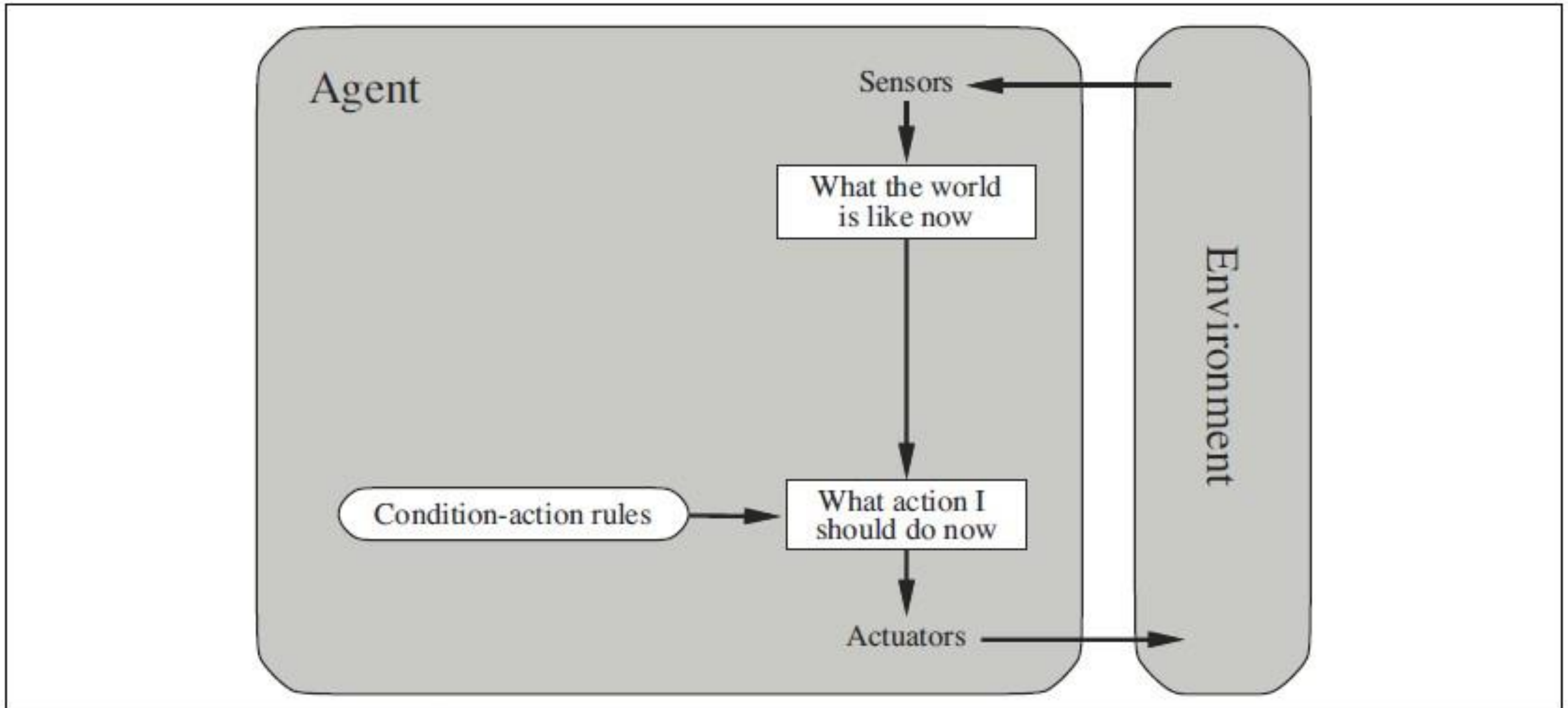


Figure 2.9 Schematic diagram of a simple reflex agent.

Agents: Russell & Norvig

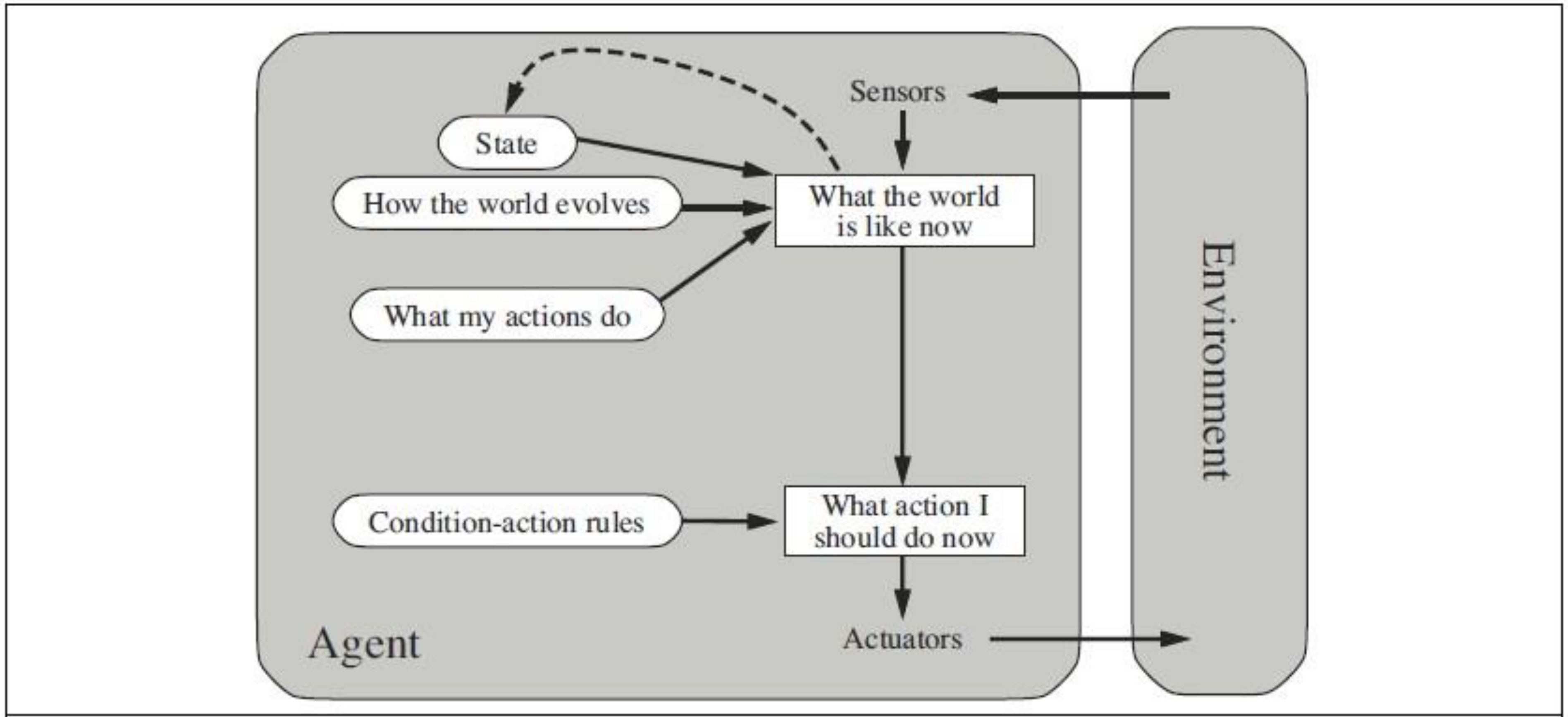


Figure 2.11 A model-based reflex agent.

Agents: Russell & Norvig

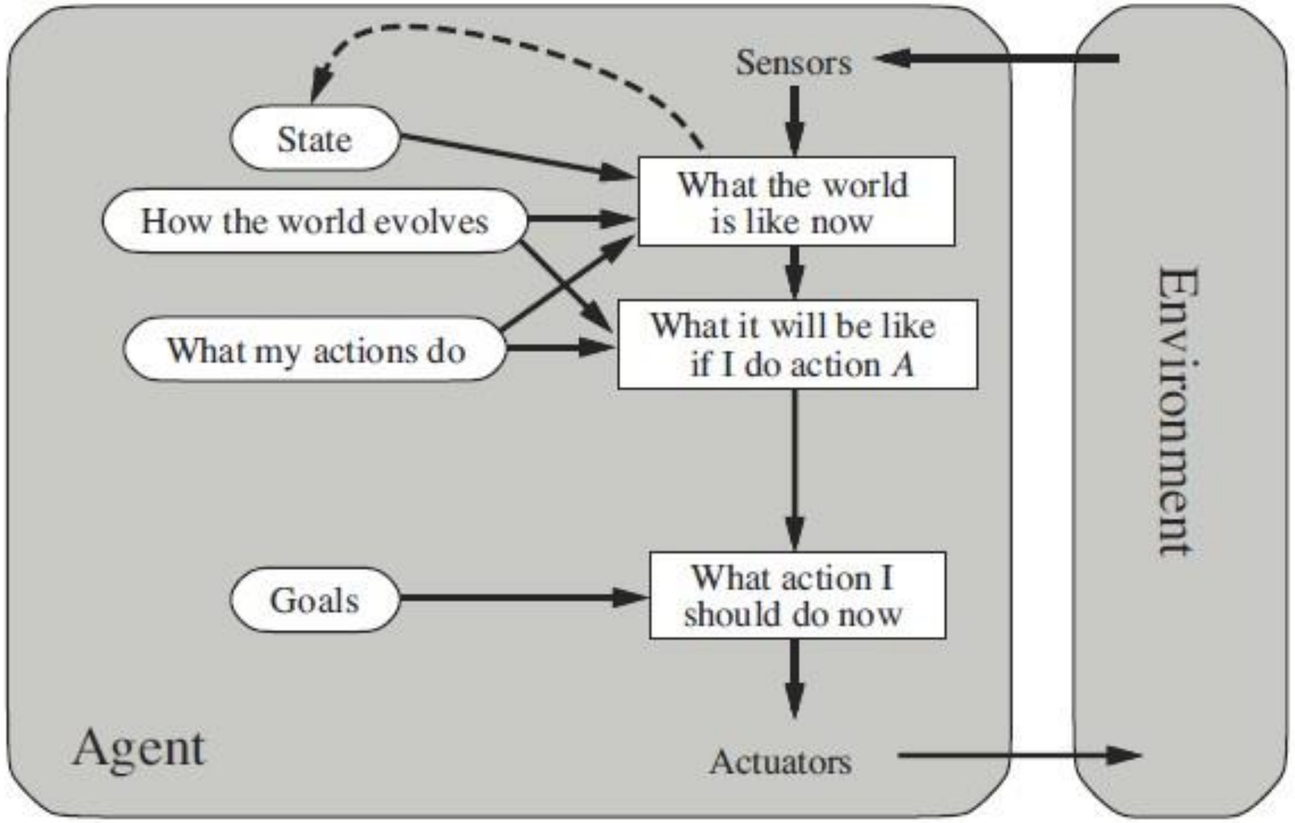

Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action *A*
What my actions do
Goals
What action I should do now
Agent
Actuators
Environment

Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Agents: Russell & Norvig

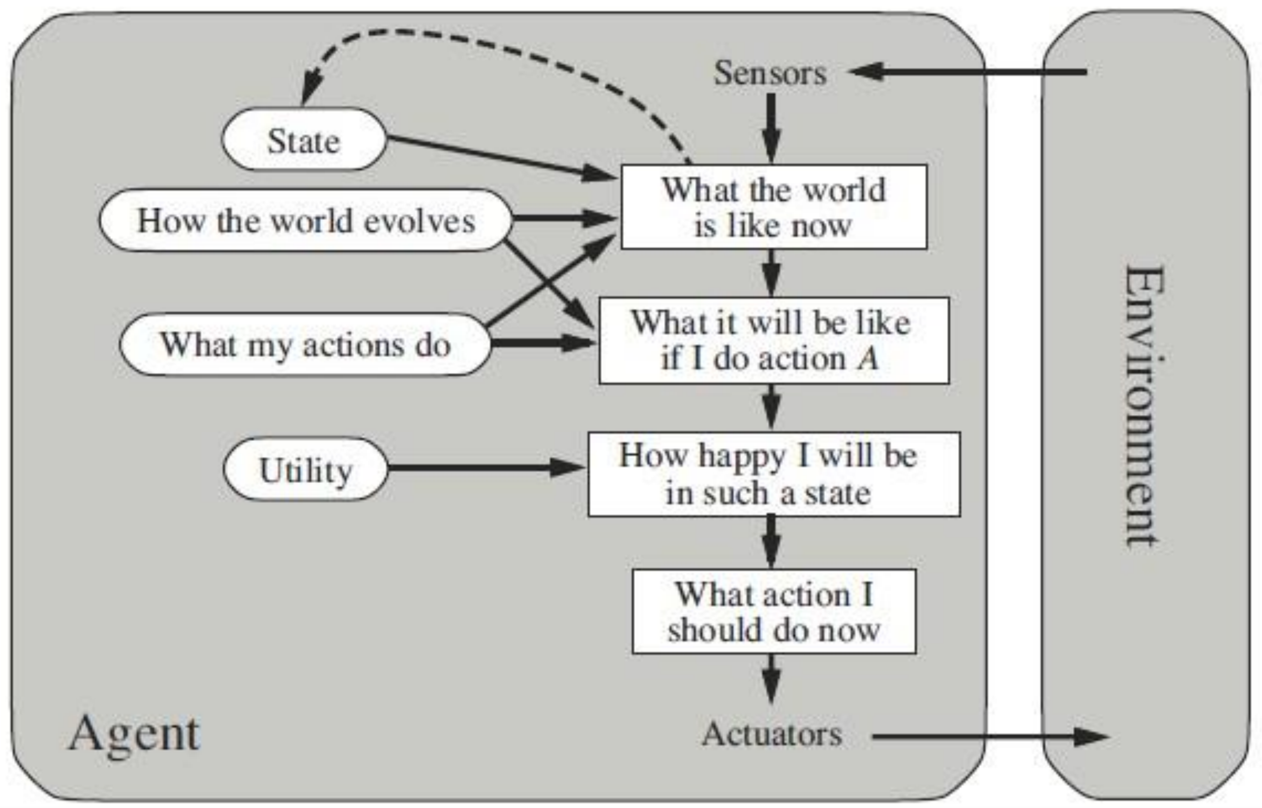

Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action *A*
What my actions do
Utility
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

Agents: Russell & Norvig

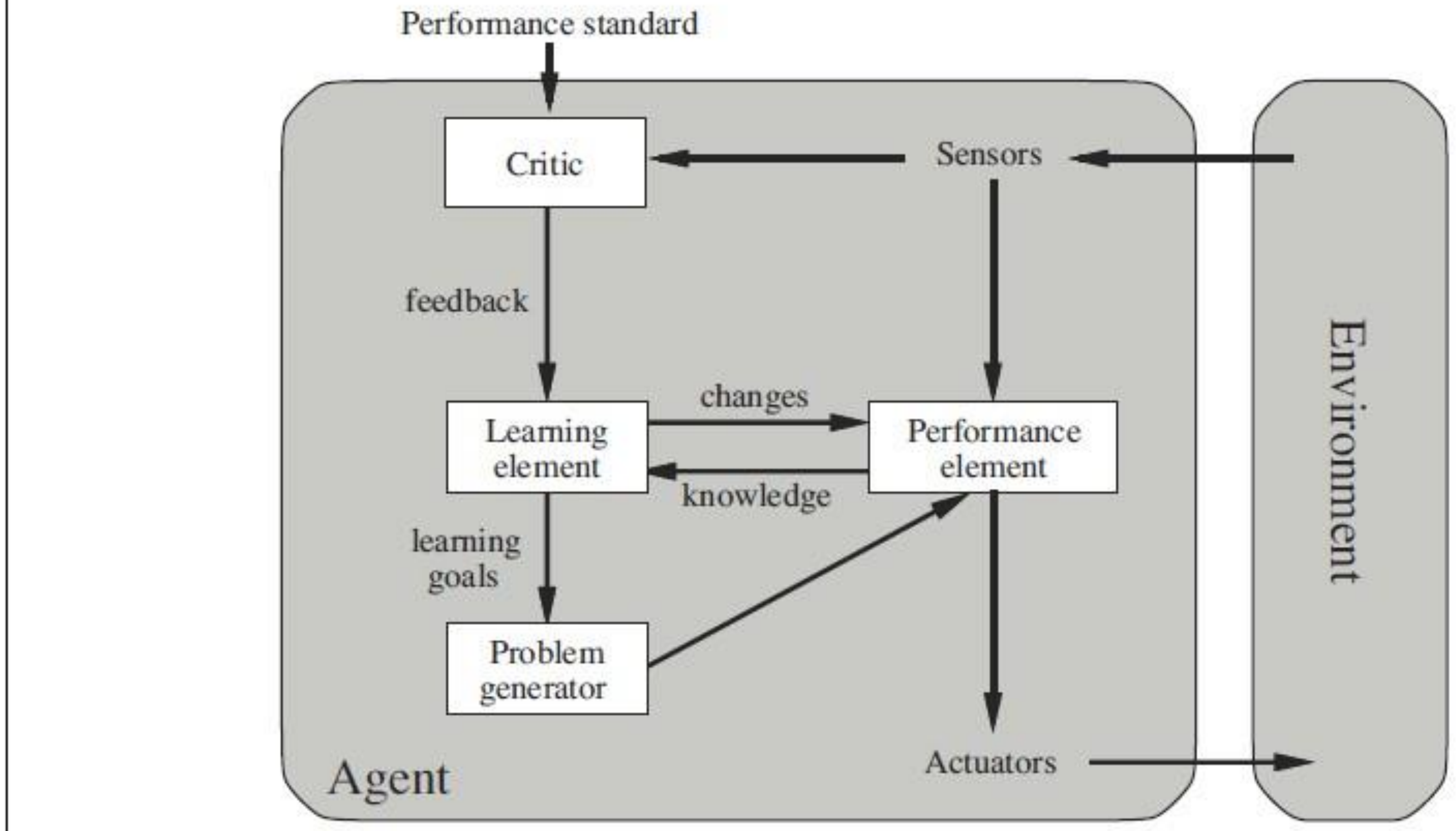


Figure 2.15 A general learning agent.