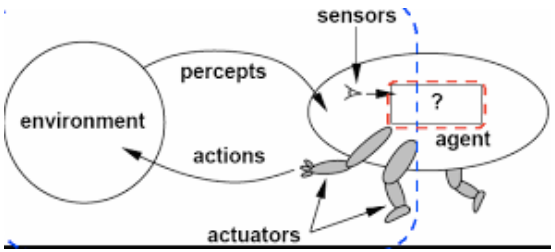
Agents
DV2557

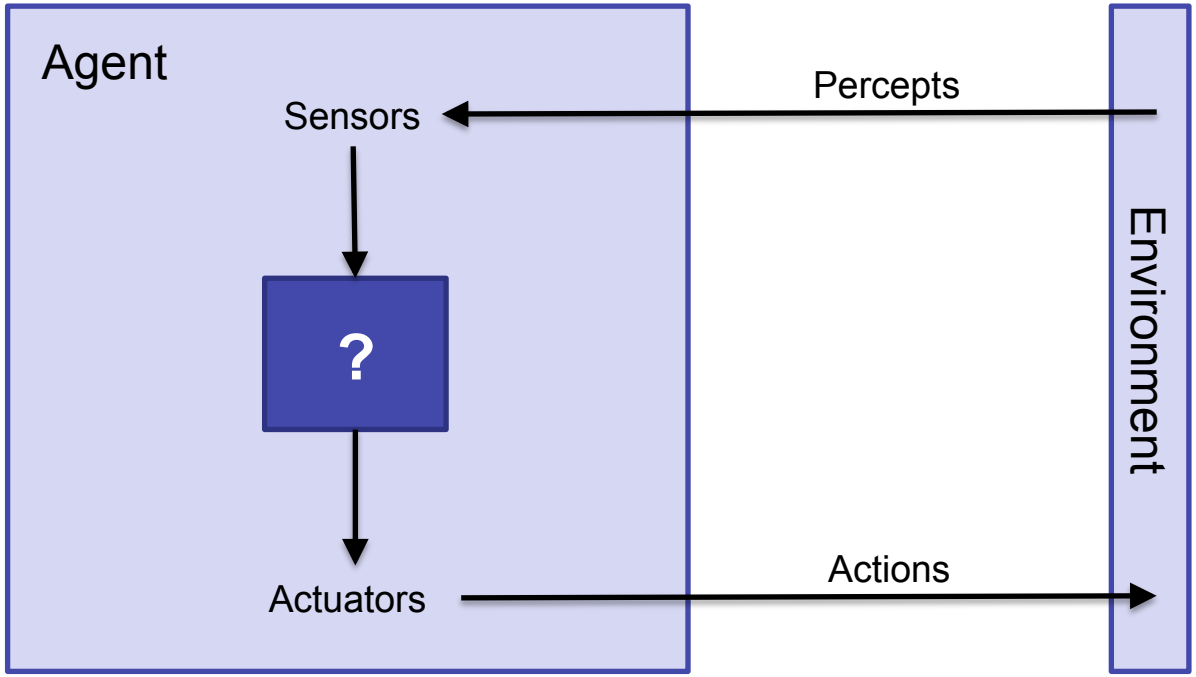
Dr. Prashant Goswami
Assistant Professor, BTH (DIKR)
https://prashantgos.github.io

prashant.goswami@bth.se



WHAT IS AN AGENT?

What is an agent?



An agent can be:

- Physical (human, animal, robot)
- Virtual (software)

Definitions

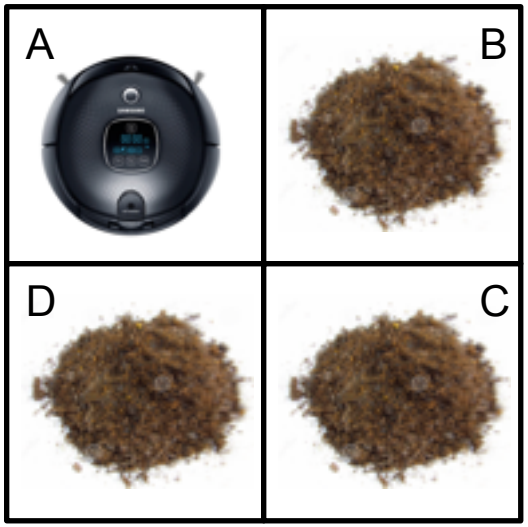
- Percept
  - The agent's input (visual, sensors, etc.) at any given instant.
  - Percept sequence if the complete history of everything the agent has perceived during its lifetime.
- Action
  - This is what the agent decides to do to manipulate the environment it is placed in.

Definitions

- Agent function
  - Maps any given percept or percept sequence to an action.
- Agent program
  - The internal program that realises the agent function.

An example: Cleaning robot

Percept Sequence	Action
[A, Dirty]	Clean
[A, Clean]	Right
[B, Dirty]	Clean
[B, Clean]	Down
[C, Dirty]	Clean
[C, Clean]	Left
[D, Dirty]	Clean
[D, Clean]	-



Good or bad behavior?

- Is the proposed actions for our cleaner robot:
  - The best possible action in every step?
  - Sufficiently good action in every/almost every step?
  - Bad in most steps?
  - Complete crap?
- Let’s introduce the concept of rationality.

The rational agent

- A rational agent is an agent that always makes the best possible action in each step.
- Of course doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing?
- We can define the best actions as the actions that makes the agent most successful during its lifetime.

Performance measures

- We need a way of measuring the performance of an agent.
- This measurement has a lowest possible value (the worst case) and a highest possible value (the best/optimal case).
- Since agents rarely reach the best case all the time, we define a threshold.
- If the agent on average is above the threshold, we consider its performance good.

Performance measures

- In some cases it is not a big deal if performance drops temporarily as long as the average is good.
  - When trading with stocks you will lose money every now and then.
- In other cases it is crucial that the performance doesn’t drop.
  - Yes, I crashed into another car. But 99% of the time I drove perfectly.

Performance measures

- It is often very complex to measure performance:
  - Different meaning of the same thing: that your house is clean will probably not be as strict as if a room for heart surgery is clean.
  - Conflicts of interests: ”yes I earn a lot of money but I work 14 hours a day and my life sucks”

Performance measures

- Lack of knowledge: Can we get all the information we need for making an accurate measurement? ”Yes this car seems to work well but I don’t know what might hide under the hood”
- Timing of feedback: In board games like Chess we don’t know if a move was good or bad until the game has ended.

Performance measures

- Noise: Is this sensor functioning correctly? Is the quality of the sensor readings sufficient?
- Average: lots of ups and downs can give the same average as a stable but not so good agent. Which is preferred?
- Reliability. Is the performance function actually measuring the correct things or is it measuring what the engineers think are the correct things?

Rationality

- What is rational at any given time depends on:
  - The agent's prior knowledge of the environment
  - The agent's percept sequence to date
  - The actions that the agent can perform
  - The performance measure that defines the criterion of success

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.

This is slightly easier to achieve than ”real” rationality. The agent has to be rational given its percepts and previous knowledge, not what is rational given perfect knowledge and perfect percepts.

Rationality vs. Omniscience

- There is a distinction between rationality and omniscience.
- Omniscience means that the agent always knows the actual outcome of its actions.
- This is in reality almost always impossible.
  - A pedestrian got hit by a meteor while crossing an otherwise empty street.
  - Was he/she acting irrational when crossing the street?

Rationality vs. Omniscience

- Rationality is not the same as perfection.
- Rationality maximizes *expected* performance.
- Perfection maximizes *actual* performance.
- We can strive to fulfill rationality, but omniscience requires rapid development of crystal balls or time machines.
  - To be sure of actual performance we need to percept the future…

Rationality

- To be able to make rational decisions the agent must have all relevant information.
- When crossing a street the agent should:
  - Look to the left for cars or bikes
  - Look to the right for cars or bikes
  - But it doesn't need to look up for meteors or crashing airplanes because these event are too unlikely to happen.
- This step is called *Information Gathering*.

Exploration

- An agent that does not have complete knowledge about its environment, like our cleaning robot, it needs to *explore*.
- If exploring shall be useful the agent must also be able to *learn* from what it perceives.
- And build an internal *model* of the environment.
- To reduce the need for exploration we can give the agent some prior knowledge about the environment.

Autonomy

- Often it is desired for agents to be autonomous: they can handle themselves without the need of input from engineers.
- An agent with prior knowledge relies to some extent on its designers, so to some extent it lacks autonomy.
- In practice, an ”autonomous agent” is not autonomous at the start but will be when it has learned enough.

ENVIRONMENT

PEAS

- When designing a rational agent we need to define the PEAS:
  - Performance measure
  - Environment
  - Actuators
  - Sensors

PEAS example

Agent Type	Performance Measure	Environment	Actuators	Sensors
Self-driving car	Safe, fast, legal, comfortable trip, minimize fuel consumption	Roads, other traffic, pedestrians, bikes, wild animals	Steering, accelerator, brake, signal, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, fuel sensor

Environment properties

- Environments have different properties that makes them more or less complex for agents.
- Of course all environments are unique, but we can define some general properties that helps us reason about the complexity of an environment.

Environment properties

- Fully vs. partially observable
  - Can the sensors give a complete view of the environment at each point in time?
  - Fully observable are easier to deal with, but are often not possible in reality.
  - The self-driving car can see other cars, but cannot see what their drivers are thinking.
  - Chess is fully observable, Poker is not.

Environment properties

- Deterministic vs. stochastic
  - If the next state of the environment only depends on the current state and the agents action, the environment is deterministic.
  - If not, it is stochastic.
  - Deterministic environments are of course easier for agents, but are often not possible in reality.
  - Driving a car is stochastic, since a lot of things can happen in traffic.
  - If the environment is deterministic except for the actions of other agents, we say that it is *strategic*.

Environment properties

- Episodic vs. sequential
  - In episodic environments the agent's execution is divided into atomic episodes.
  - In each episode the agent receives percepts, and carry out an action.
  - The choice of action in each episode only depends on the episode itself, not previous episodes.
  - One such example is a "Mail-sorting" or "Cleaning" robot, which only is interested in the current part.
  - In sequential environments the current decision could affect all future decisions.

Environment properties

- Static vs. dynamic
  - If an environment can change without it being modified by an agent, it is dynamic.
  - Static environments only change if an agent executes an action.
  - If the environment itself does not change, but the performance score measurement of the agent, we call it *semidynamic*.

Environment properties

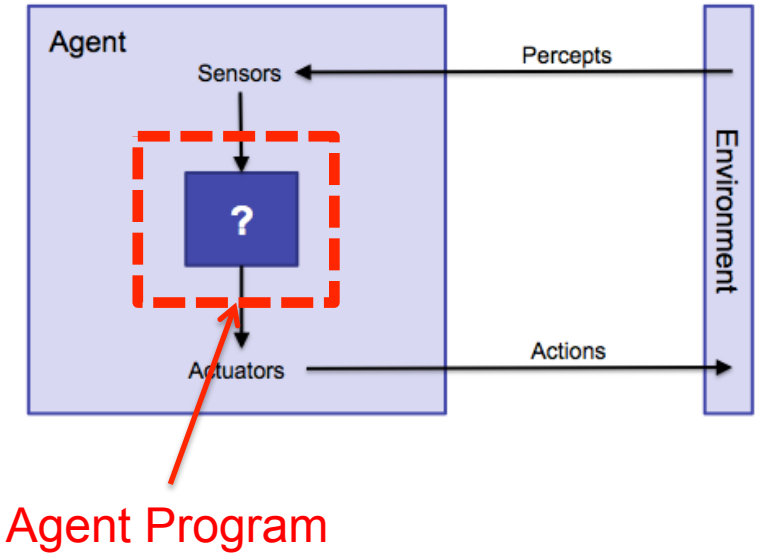
- Discrete vs. continuous
  - Discrete/continous can be applied to how time is handled, and to the percepts and actions of an agent.
  - Chess has a discrete set of percepts and actions.
  - Driving is continuous in state and in time. The sensors give a continuous stream of data and for example steering angles is real-valued.
  - Continous in time is what we often call real-time.

Environment properties

- Single agent vs. multiagent
  - Is the agent alone in the environment, or is it inhabited by other agents?
  - Multiagent environments can be competitive (agents compete about solving tasks), cooperative (agents cooperate to solve tasks) or a combination.
  - In multiagent environments communication between agents is often needed.

Environment examples

Task environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess	Fully	Deterministic	Sequential	Static	Discrete	Multi (2)
Poker	Partially	Stochastic	Sequential	Static	Discrete	Multi (2)
Driving	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical diagnosis	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
District Heating Optimization	?	?	?	?	?	?

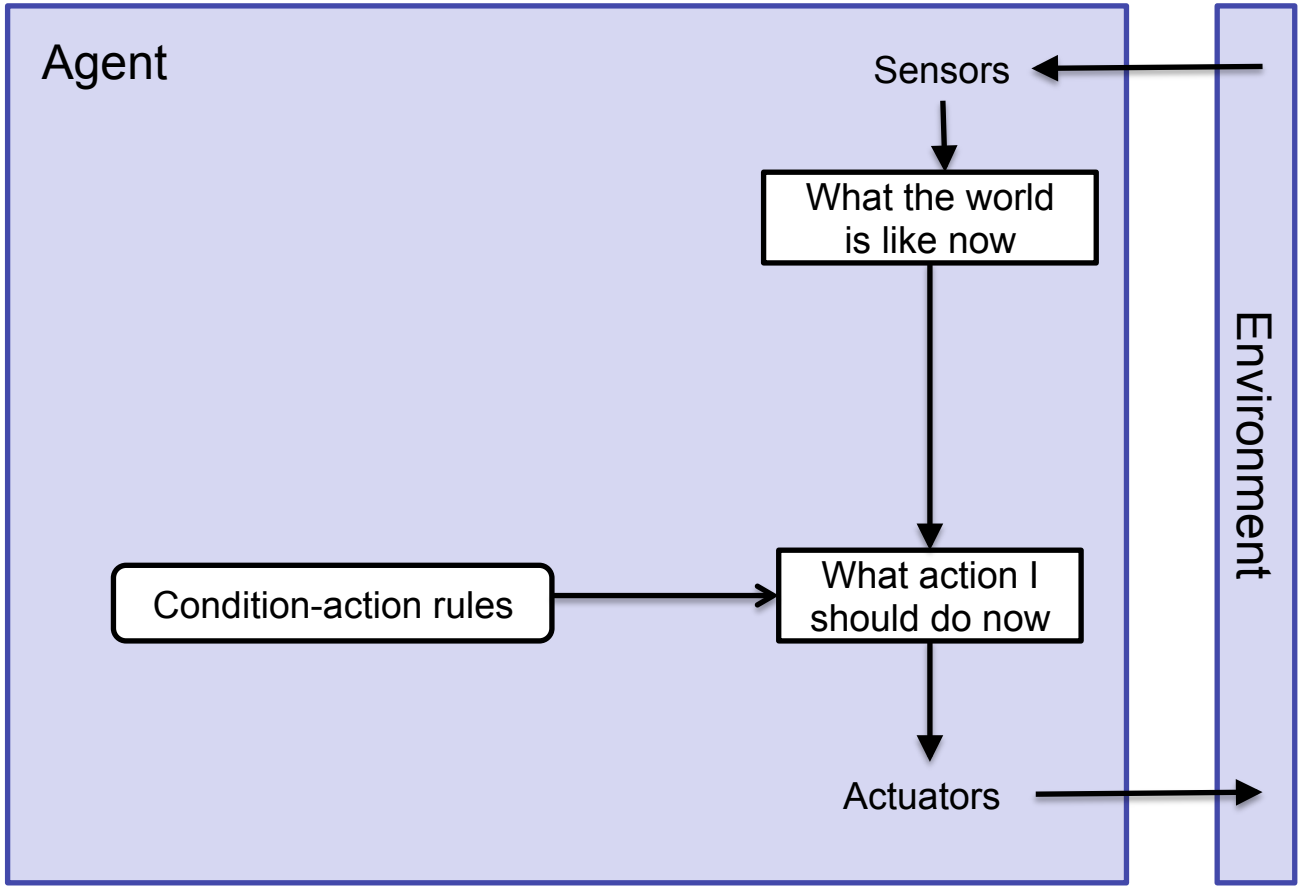


TYPES OF AGENTS

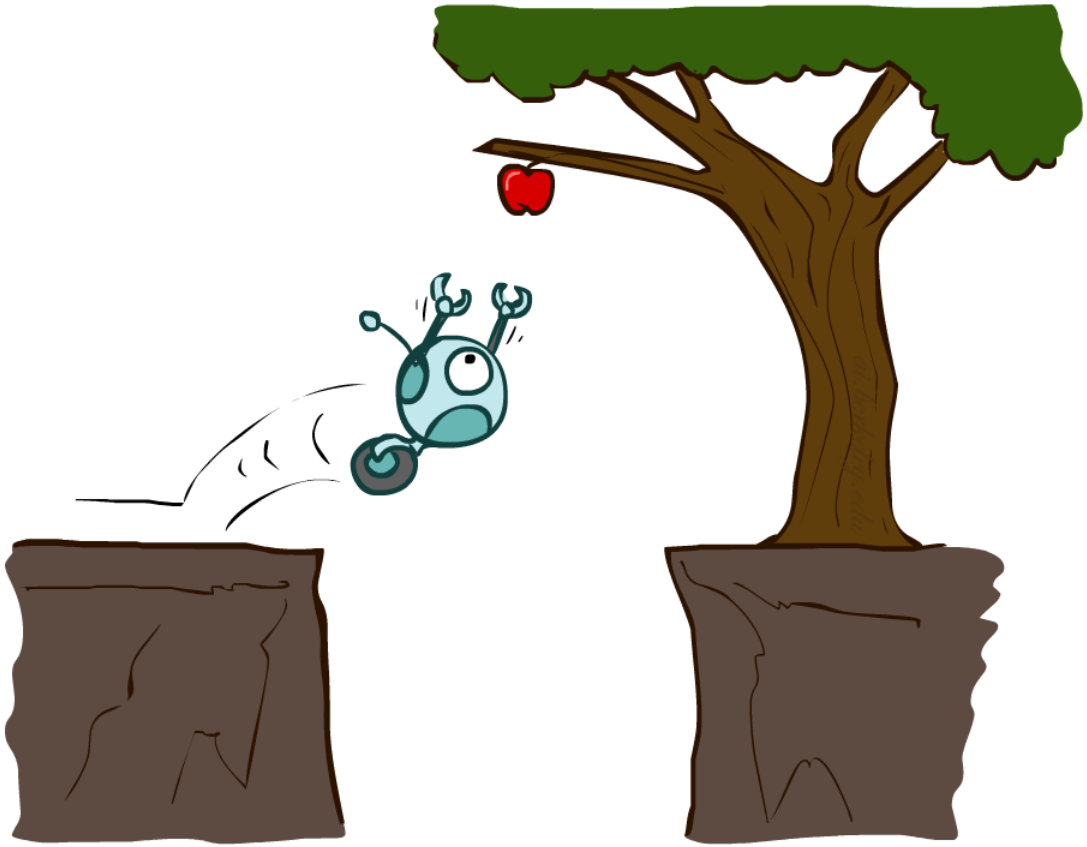
Simple reflex agent

- The simple reflex agent selects actions based only on the current percept, ignoring the rest of the percept history.
- It is the simplest agent type.
- It implemented using *condition-action rules* (*if-then rules*):
    - if *Dirty* then *Clean*
    - if *Clean* then *Right*
    - *…*

Simple reflex agent



Simple reflex agent

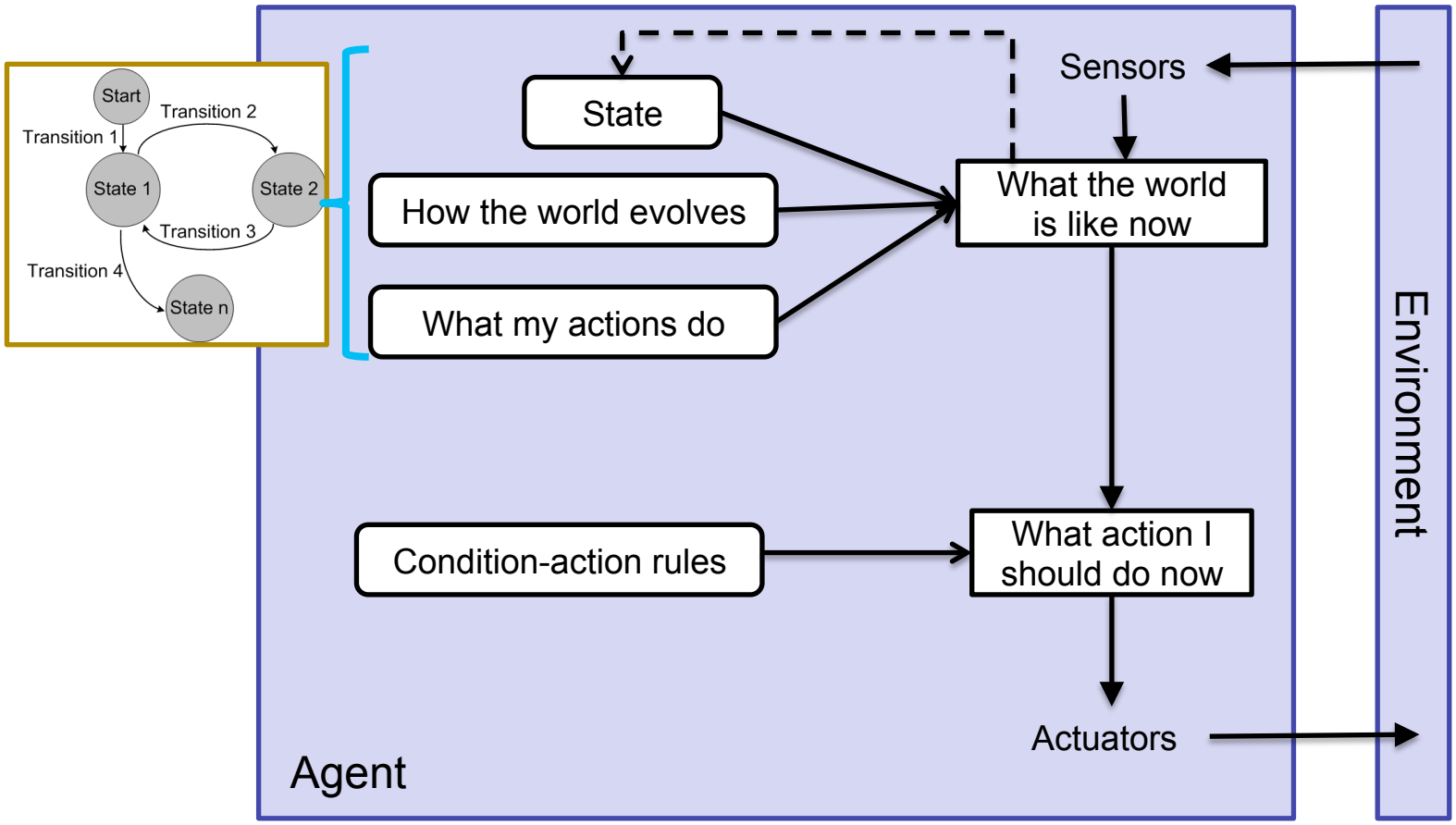


- Choose action based on current percept
- Do not consider the future consequences of their actions
- May or may not have a model of the world's current state

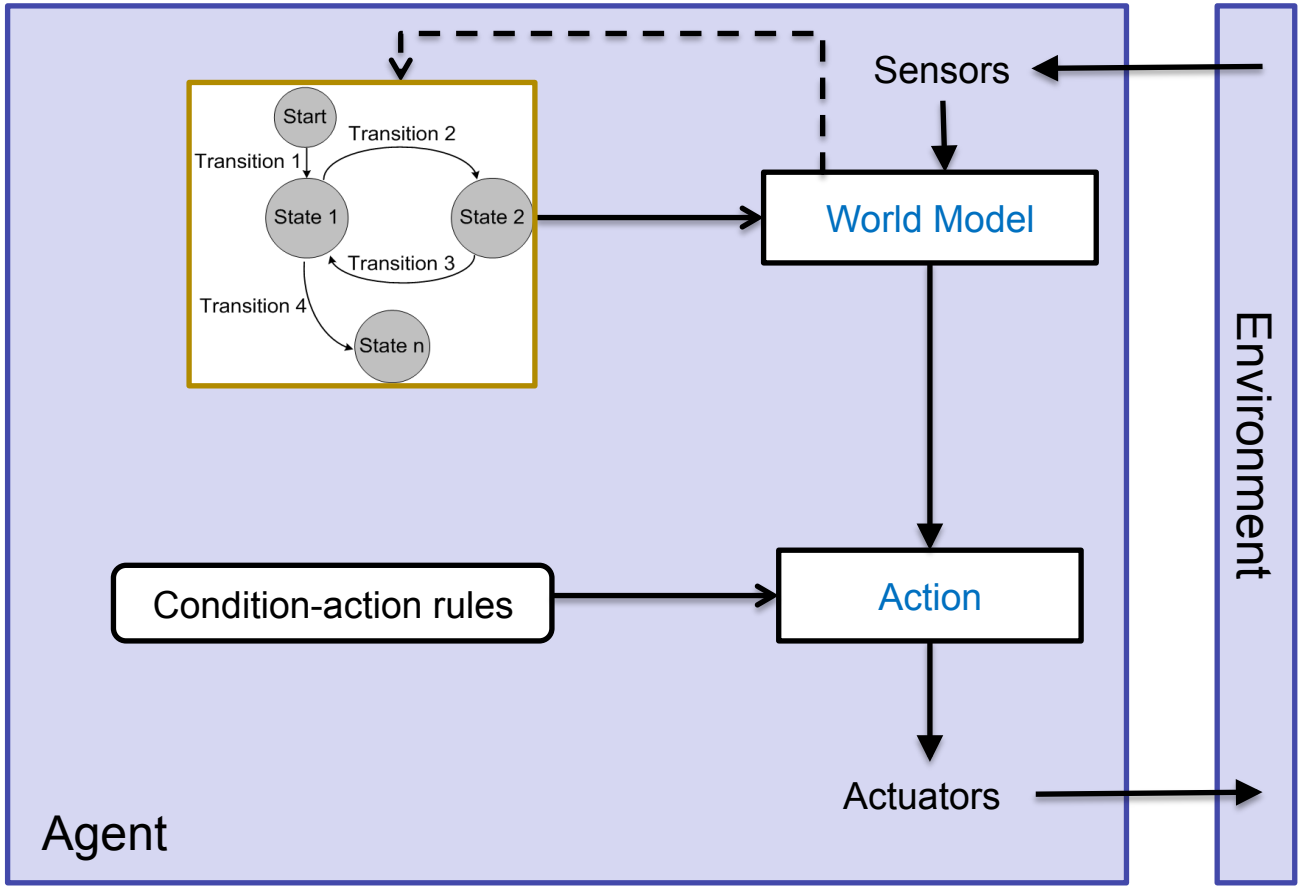
Model-based reflex agent

- The model-based reflex maintains an *internal model* of what is known about the environment.
- This is based on the percept history.
- By doing this the agent can keep track of the part of the environment it can't see now.
- It should also know how the world evolves and the effect the agents actions have.
    - Where would I be in the world if I turn right in this crossing and drive for 5 km?

Model-based reflex agent



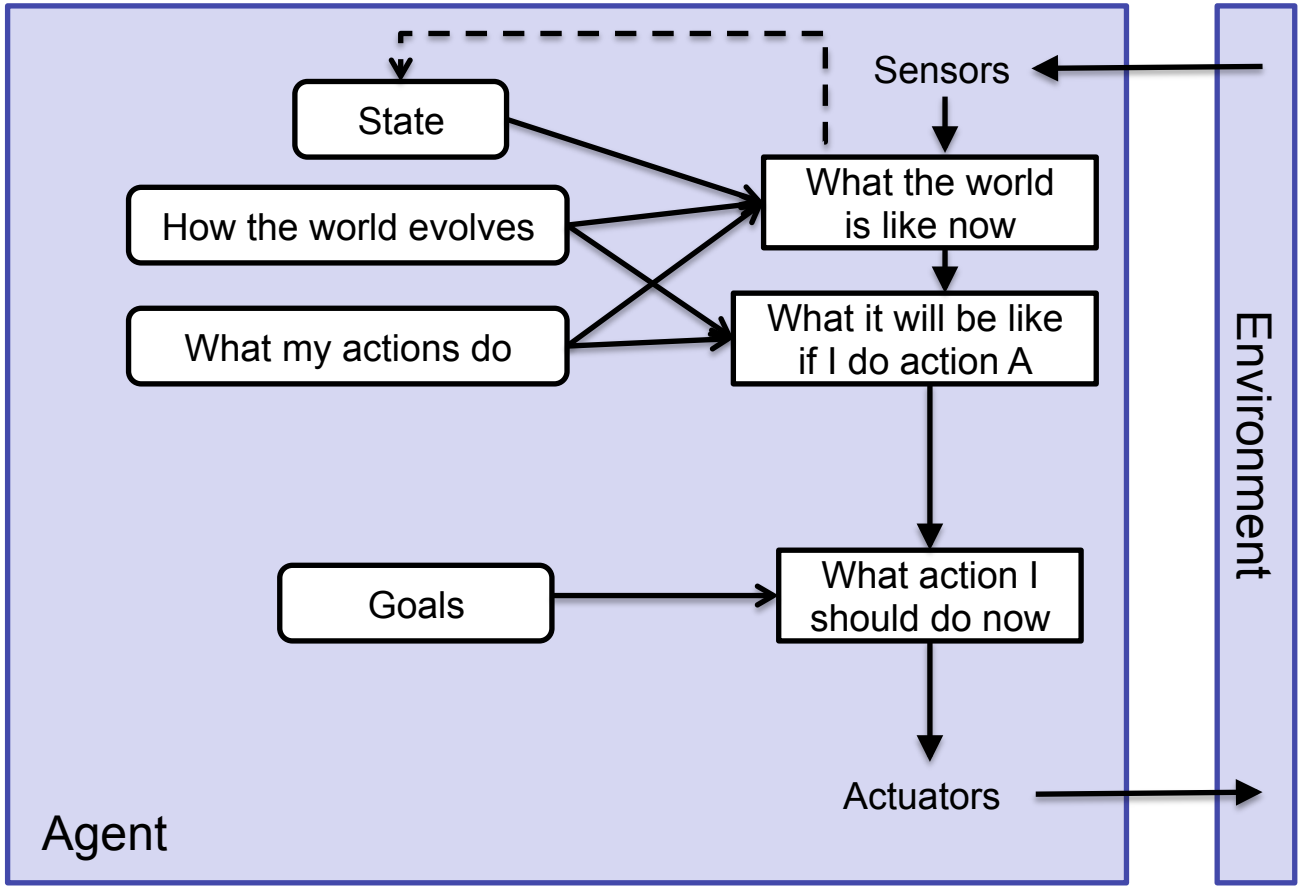
Model-based reflex agent - Simplified



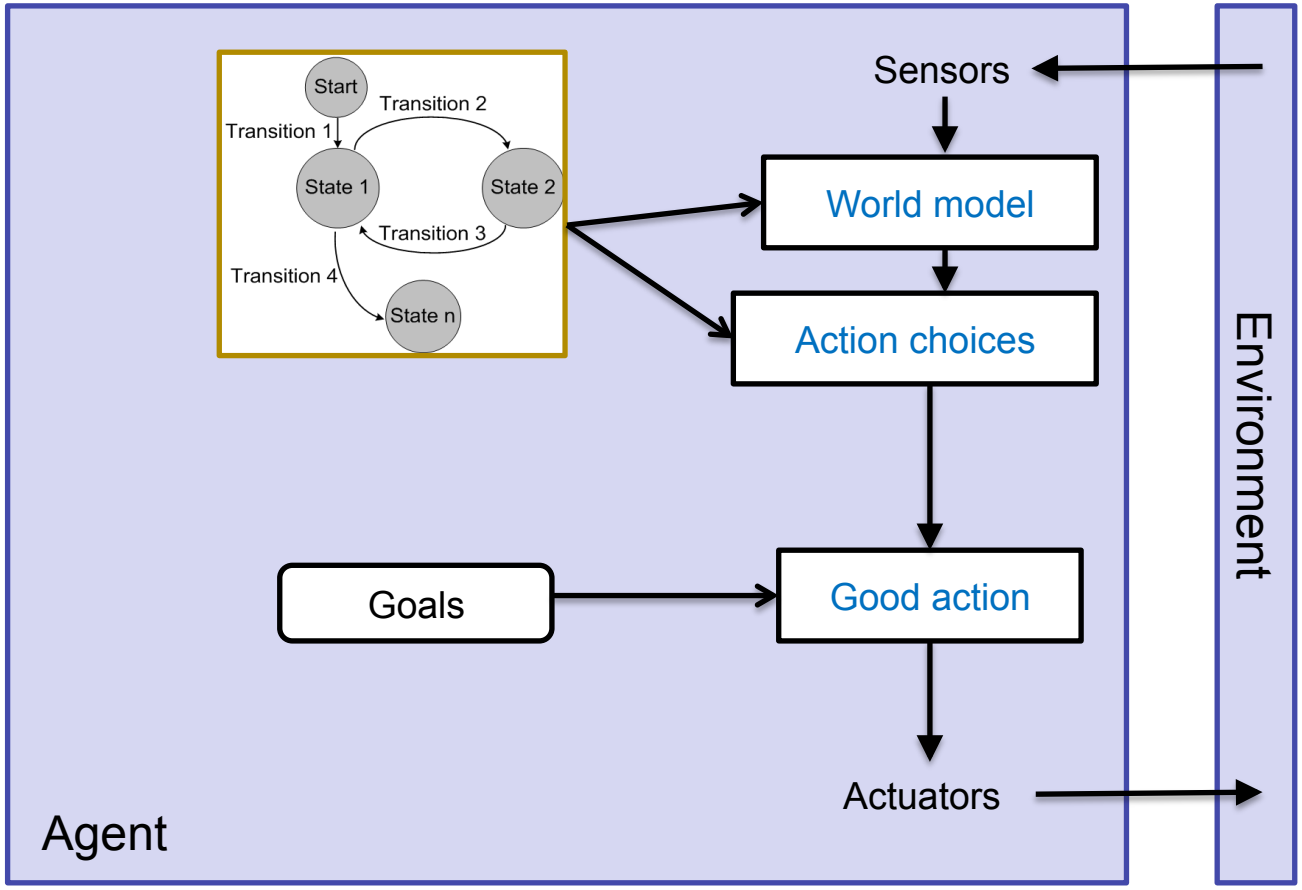
Goal-based agent

- The goal-based action execute actions that leads it closer to specific goal(s).
- To reach a goal task the agent must divide it into smaller subtasks.
- This is done using Planning techniques, which we will return to in a future lecture.

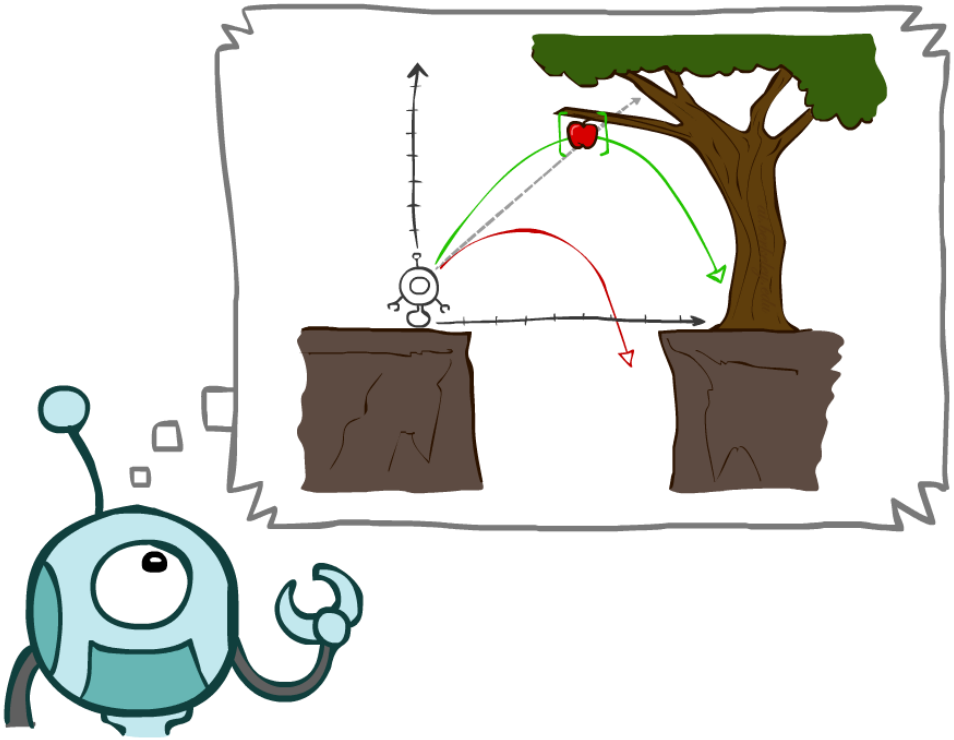
Goal-based agent



Goal Agent - Simplified



Goal-based agent

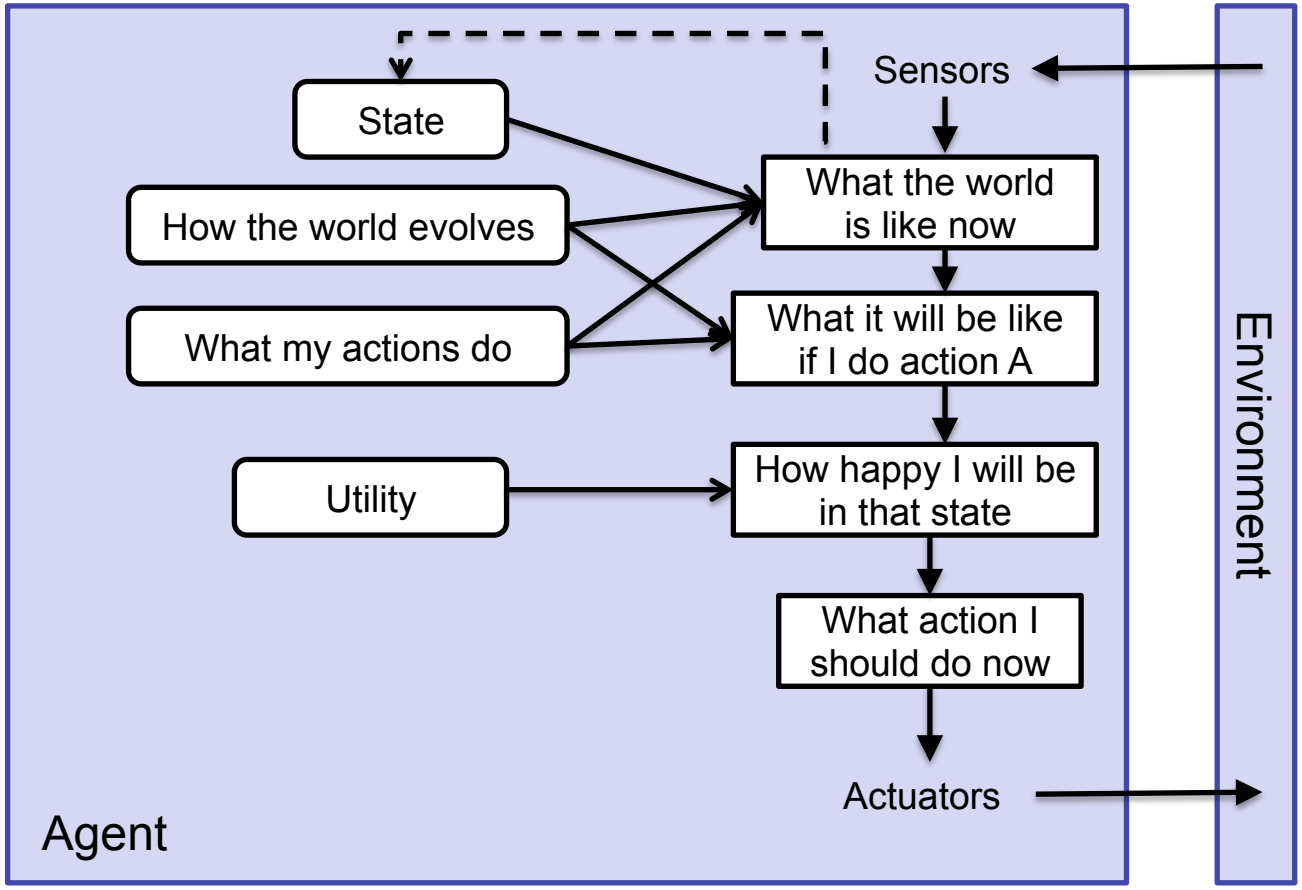


- Choose action based on current percept and world's internal model
- Choose the action that brings it closer to goal

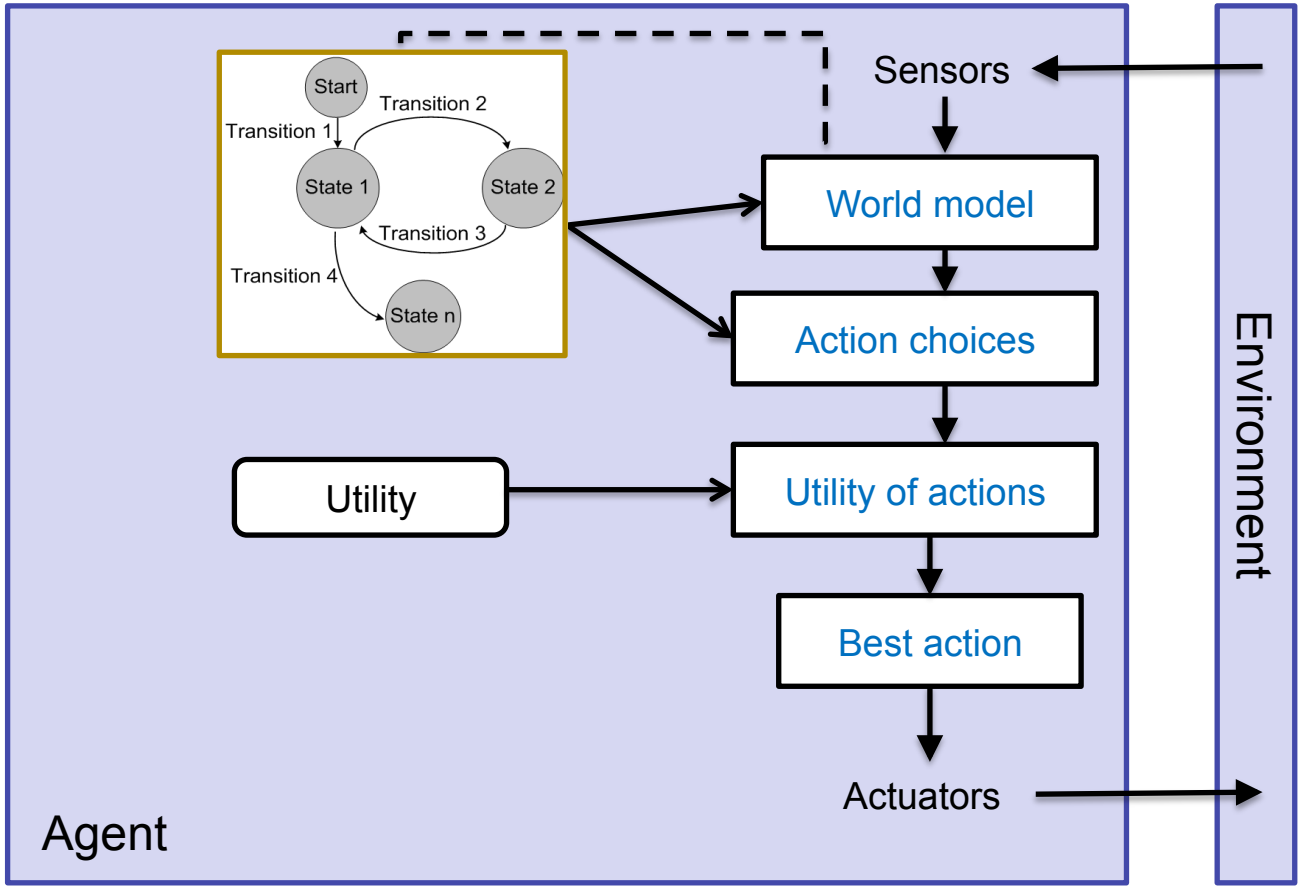
Utility-based agent

- The utility-based agent uses an utility-function.
- This function gives a value of how desirable (good or bad) a state is.
- The agent choses the action that leads to the state with the highest utility.
- One example is Chess bots, which we will talk about in next lecture.

Utility-based agent



Utility-based agent



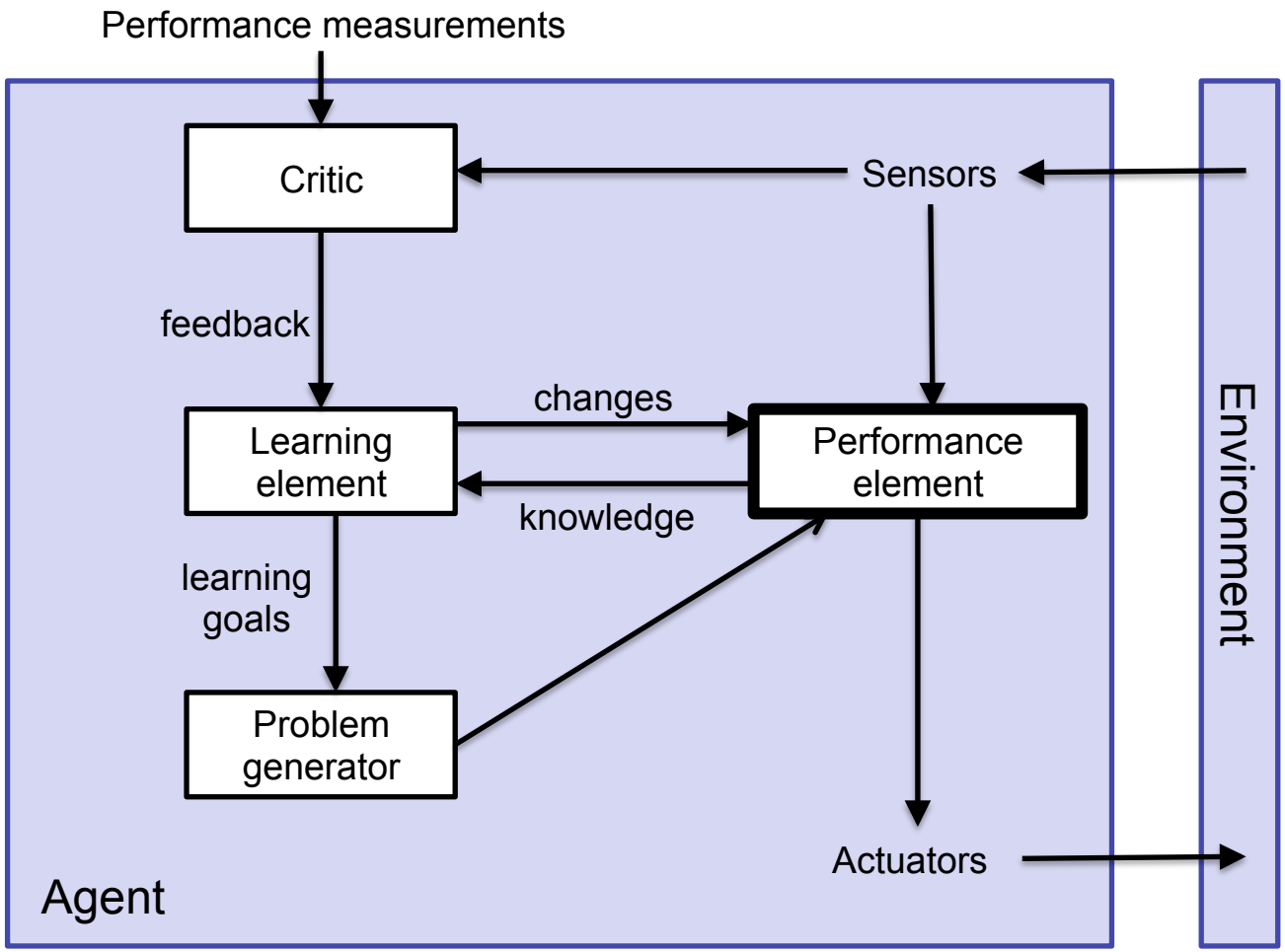
Utility-based agent

- Utility-based agent is more flexible
  - Chooses best out of several goals
  - Appropriate trade-off for conflicting goals
- Partial observability and stochasticity ubiquitous in real world

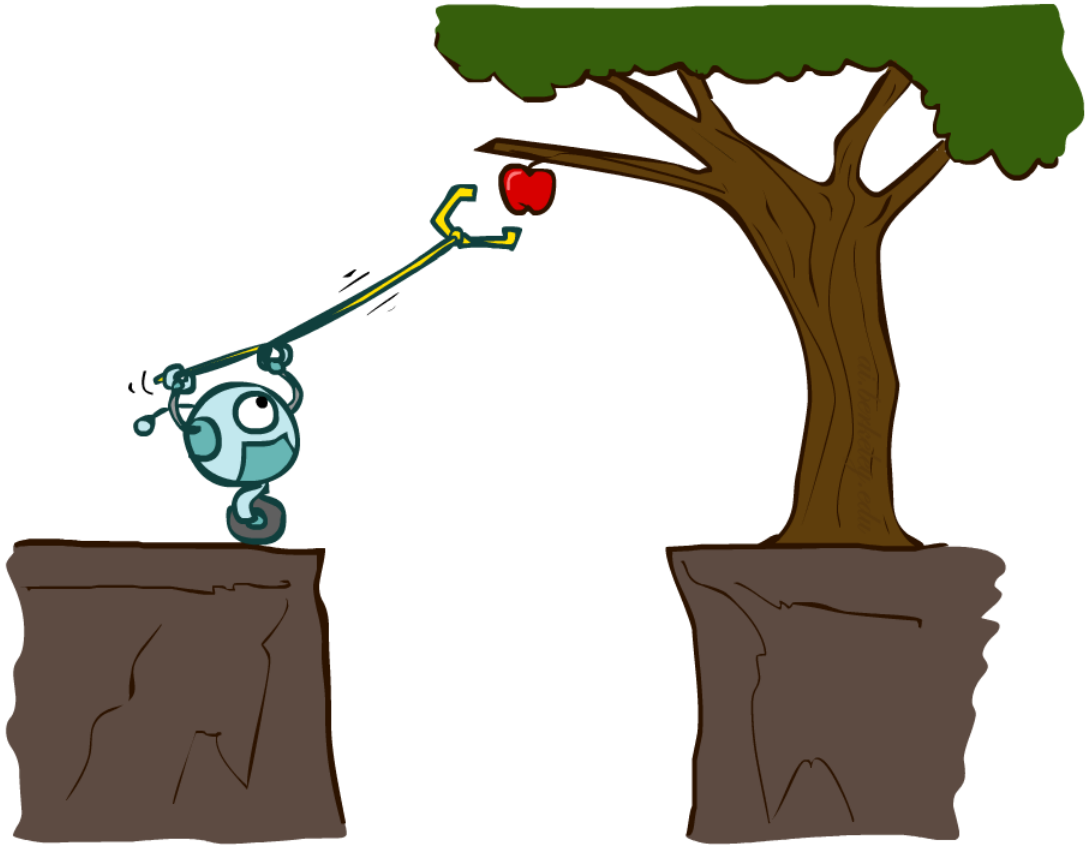
Learning agent

- The learning agent is able to learn from what it percepts.
- It has four conceptual parts:
  - Learning element: responsible for making improvements.
  - Performance element: responsible for selecting actions.
  - Critic: gives feedback to the learning element on how the agent is doing.
  - Problem generator: responsible for suggesting actions that will lead to new experiences and information.
- The agent has to balance exploration (trying out new things to learn more) and exploitation (doing what it thinks is the best in a situation).

Learning agent



Advanced Agents



- Ask ”what if”
- Decisions based on consequences of actions
- Must have a world model
- Must formulate a goal

CHOOSE THE MINIMALLY BEST AGENT FOR

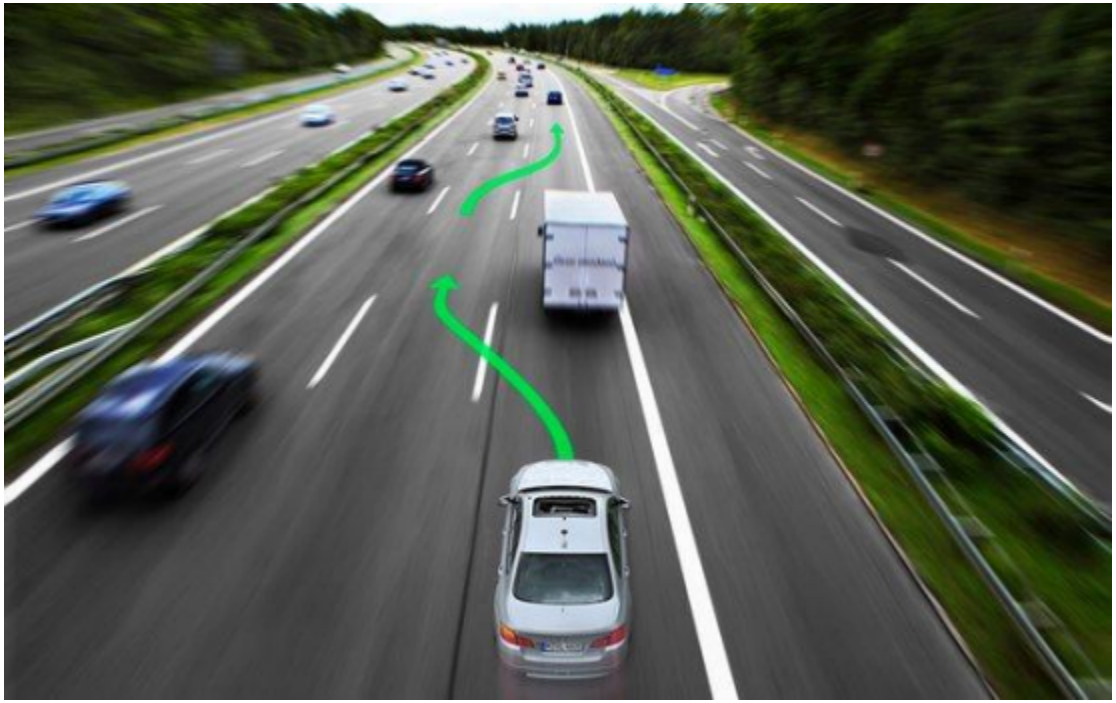
Which agent?

Simple reflex agent

Which agent?

Goal-based agent

Which agent?

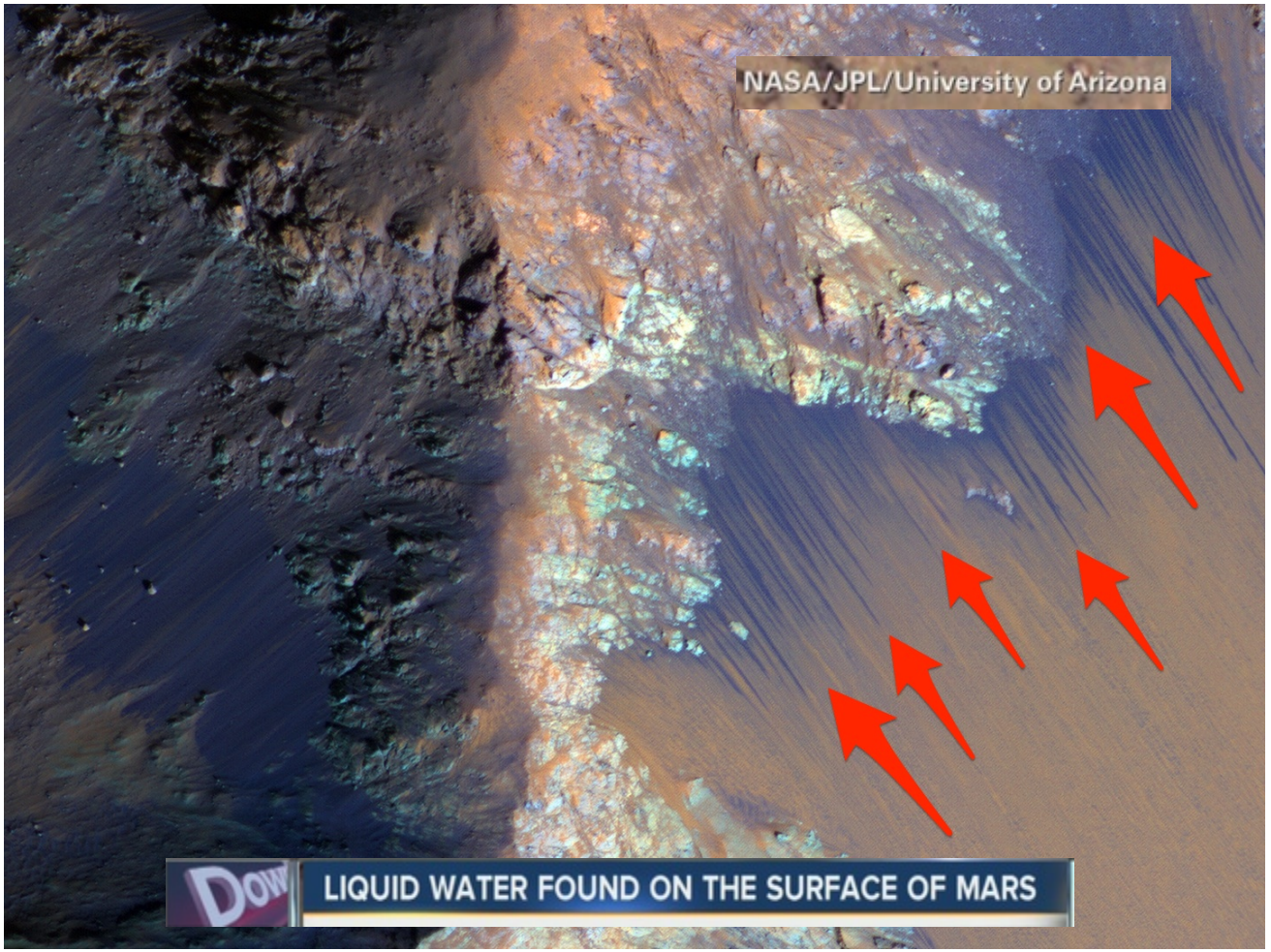


Model-based reflex agent

Which agent

Utility-based agent

Which agent?



Learning agent

EXPERT SYSTEMS

Expert Systems

- The reflex agent uses condition-action (if-then) rules.
- A rule-based knowledge and decision making system is what we usually call an *expert system*.
- The idea is that the system shall mimic a human expert by having extensive knowledge about an area, therefore the name.
- Expert systems were among the first successful AI systems.
- The first systems appeared during the 1970's and they were very popular in the 80's.

Expert Systems

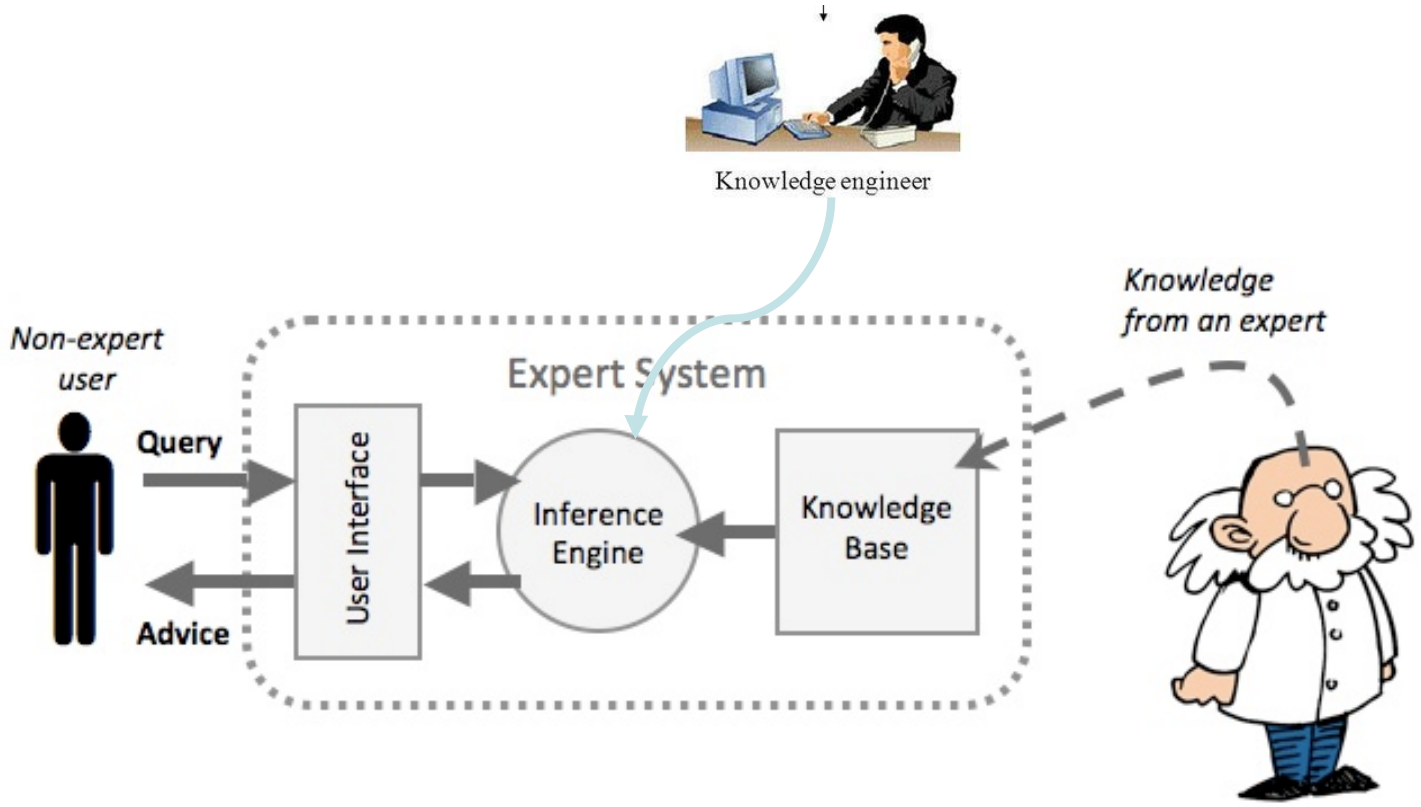


Image Courtesy: https://sjhobbs.wikispaces.com/Robotics,+AI+and+Expert+Systems

Expert Systems

Evolution of Expert Systems Software

- Expert system shell
  - Collection of software packages & tools to design, develop, implement, and maintain expert systems

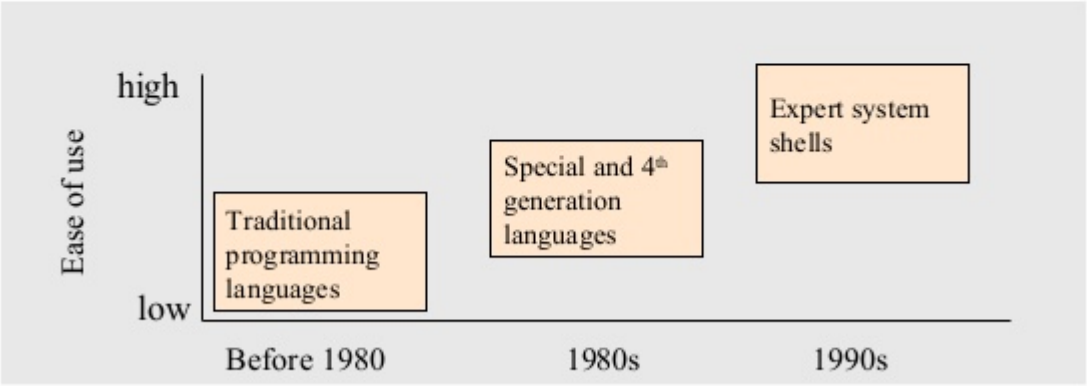
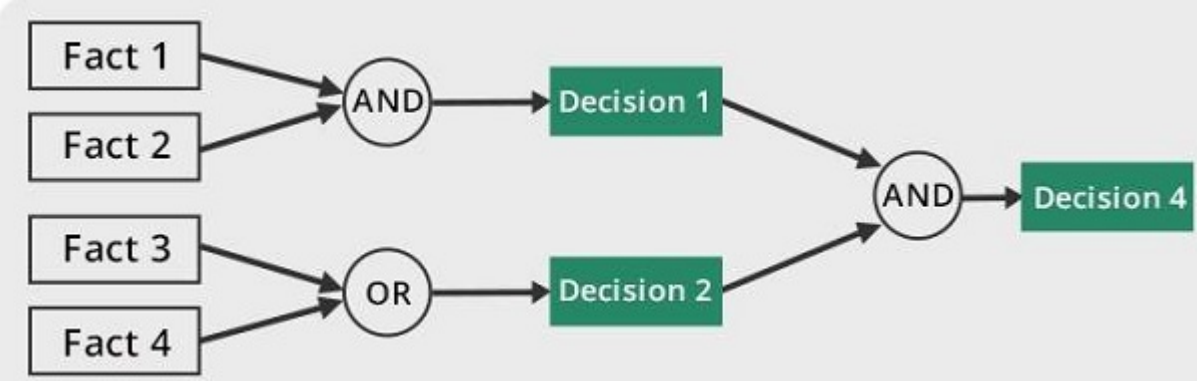


Image Courtesy: York University

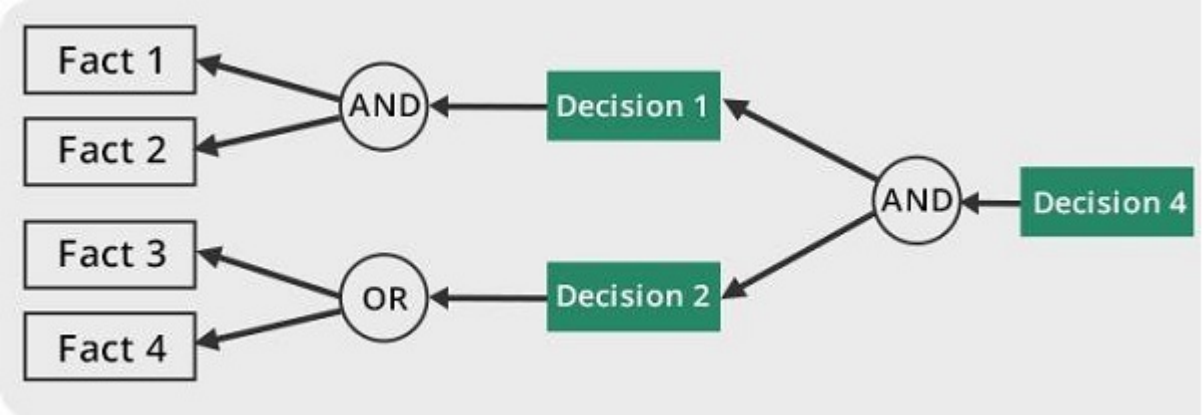
Expert Systems

- Most expert systems start with observations/facts and apply rules to reach a conclusion.
- This is called a Forward-Chaining system.
- It is also possible to go from one observation, a cheetah, and derive facts about it.
    - Backward-Chaining system

Forward- vs. Backward Chaining



Forward



Backward

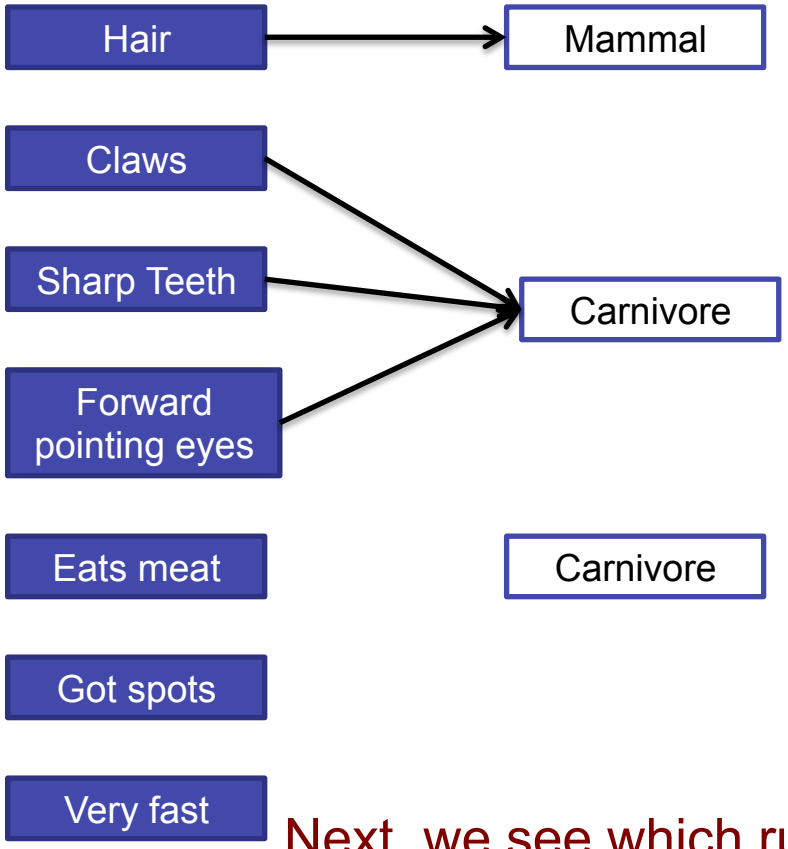
Expert System example

- Hair
- Claws
- Sharp Teeth
- Forward pointing eyes
- Eats meat
- Got spots
- Very fast

R1	IF	the animal has hair
	THEN	it is a mammal
R2	IF	the animal gives milk
	THEN	it is a mammal
R3	IF	the animal has feathers
	THEN	it is a bird
R4	IF	the animal flies
		the animal lays eggs
	THEN	it is a bird
R5	IF	the animal is a mammal
		the animal eats meat
	THEN	it is a carnivore
R6	IF	the animal is a mammal
		the animal has pointed teeth
		the animal has claws
		the animal's eyes point forward
	THEN	it is a carnivore
R7	IF	the animal is a mammal
		the animal has hooves
	THEN	it is an ungulate
R8	IF	the animal is a mammal
		the animal chews cud
	THEN	it is an ungulate AND
		it is even-toed
R9	IF	the animal is a carnivore
		the animal has a tawny colour
		the animal has dark spots
	THEN	it is a cheetah

We start by observing facts.

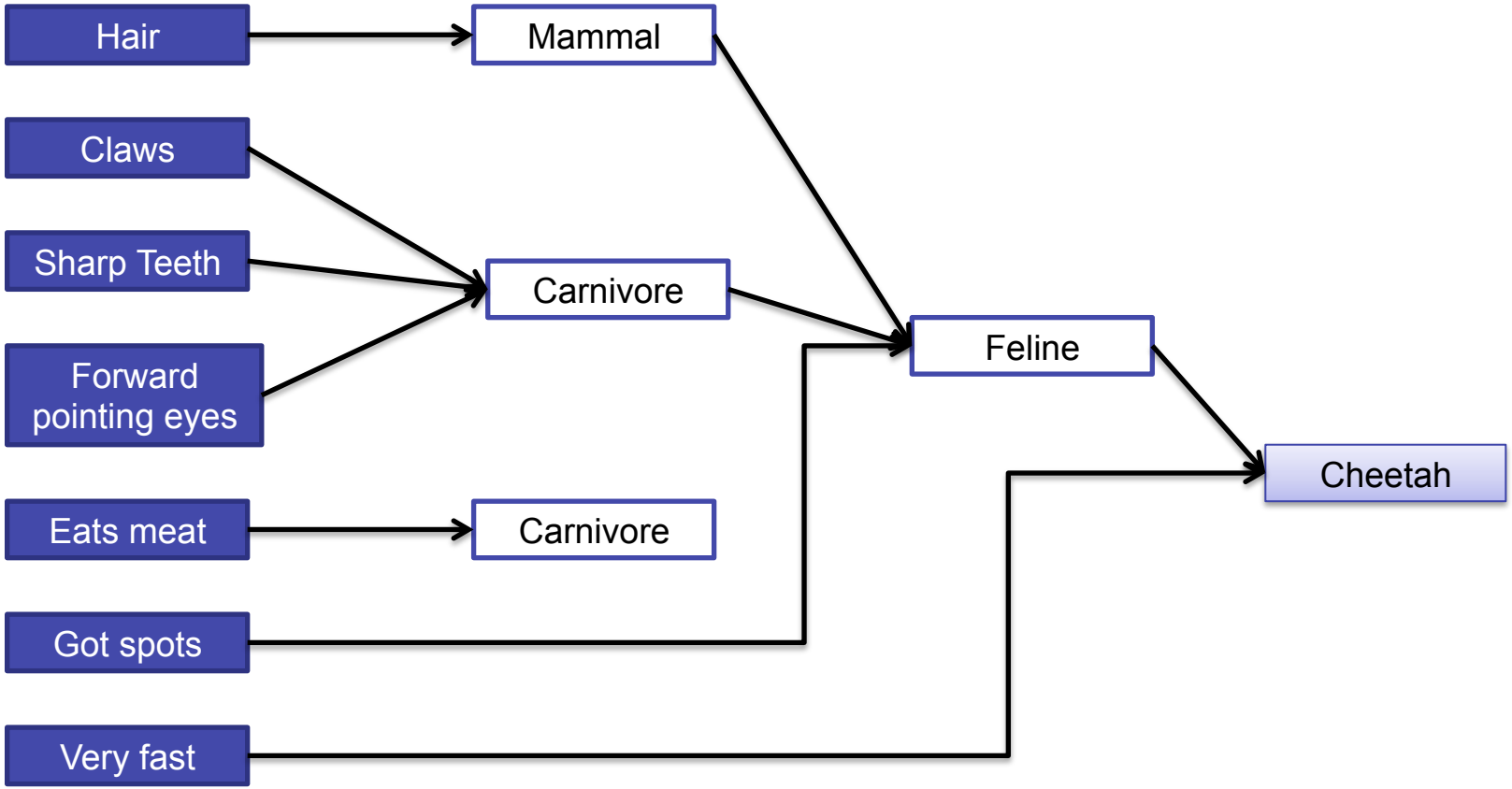
Expert System example



R1	IF	the animal has hair
	THEN	it is a mammal
R2	IF	the animal gives milk
	THEN	it is a mammal
R3	IF	the animal has feathers
	THEN	it is a bird
R4	IF	the animal flies the animal lays eggs
	THEN	it is a bird
R5	IF	the animal is a mammal the animal eats meat
	THEN	it is a carnivore
R6	IF	the animal is a mammal the animal has pointed teeth the animal has claws the animal's eyes point forward
	THEN	it is a carnivore
R7	IF	the animal is a mammal the animal has hooves
	THEN	it is an ungulate
R8	IF	the animal is a mammal the animal chews cud
	THEN	it is an ungulate AND it is even-toed
R9	IF	the animal is a carnivore the animal has a tawny colour the animal has dark spots
	THEN	it is a cheetah

Next, we see which rules fron our knowledge that can be applied.

Expert System example



And we keep doing this until a solution is found.

Limitations

- Expert Systems know what rules to apply, not when to apply it.
- We need to feed them with fact each time we want an answer – not really intelligent.
- There are always special cases, and the rule base needs to be updated.
- The experts have the knowledge but aren't engineers, the engineers can code new rules but don't have the knowledge.
- This limitation is solved by using rule languages like Logic. We will learn about this in a future lecture.

AGENT APPLICATIONS

Simulation

- Agents can be placed in virtual environments to simulate different real-world problems.
  - Traffic in cities
  - Economics
  - District heating systems
  - Evacuation of buildings
  - …
- Lets take a look at an example.

The Agent City

- Agent City is a virtual city populated by three types of agents.
- Car brands are very important for the agents.
- Agents don’t like if their neighbours have a car brand they dislike.
- Now, lets simulate how segregation in cities work.

The different agents

- Blue
  - Like safe and robust Swedish cars: Volvo or Saab.
- Red
  - German engineering is the best: Mercedes, BMW or Audi.
- Green
  - American cars are good value for the money, especially Ford.

The simulation

- Each month an agent takes a look at their closest neighbors (5x5 tiles) to evaluate how good their neighborhood is.
  - Calculate a neighborhood score
  - +0.3 for each neighbor with the “correct” car brand
  - -0.4 for each neighbor with the “wrong” car brand
  - Score range from -9.6 to 7.2
  - (we dislike bad things slightly more than we like good things)

The simulation

- The agent has a chance of moving to a new place based on the score:

Score	Chance to move
>= 0	0%
>0 to -1	0.5%
-1 to -2	1%
-2 to -3	2%
< -3	6%

- An agent only consider moving to spots with higher score, and chooses the best spot available.
- An agent always stays at least 6 months before considering to move.

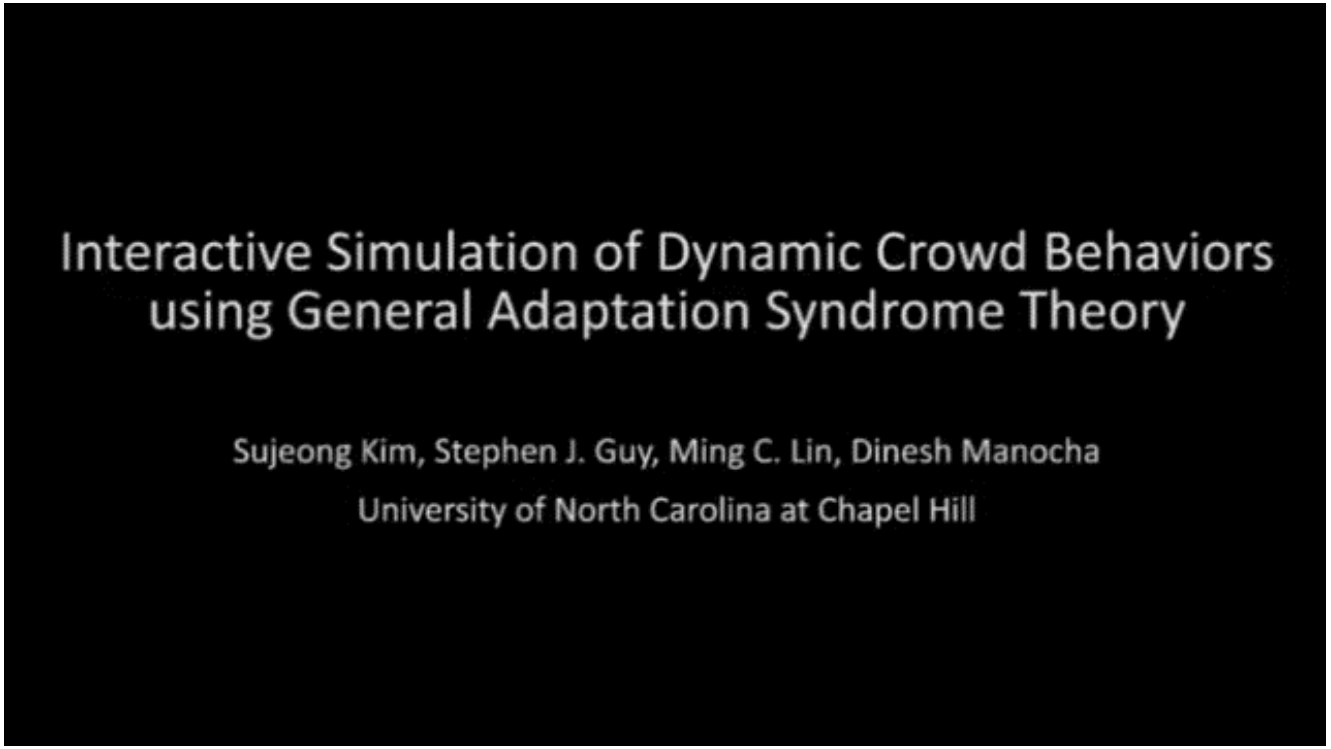
Simulation Tool

- The simulation tool is available here: http://aiguy.org/AgentCity.html

Representative to the real world?

- Of course this simulation is very simplified to the real world.
- There are more rules that apply, for example economic losses/gains of moving.
- Almost all simulations are more of less simplified compared to their real world counterparts.
- It has however been proved several times that a simplified simulation can be quite accurate!
    - Especially in short terms, well restricted problems.

Agents - Movie



Emergent Behavior

- A number of very simple agents are placed in an environment.
- When interacting with each other and the environment, all agents as a whole can form very complex behaviors.
- The complex behavior is what we call an emergent behavior (or emergent property).
- Let’s take a look at an example.

Flocking: Boids

- Boids is an algorithm for simulating flocking, for example a flock of birds or school of fish.
- It was first published by Craig Reynolds in 1986.
- Several agents (often called boids) are placed in an environment.
- Each agent/boid follows some simple rules:

Boids algorithm

1. Move towards the center (average position) of all agents.
   - Rule of Cohesion
2. Steer towards the average direction/heading of all agents.
   - Rule of alignment
3. Steer to avoid colliding into nearby agents or obstacles.
   - Rule of separation
4. Steer towards a goal (optional).

Pseudocode

```
initialise_positions()

LOOP
    draw_boids()
    move_all_to_new_positions()
END LOOP
```

Separation:
Steer to avoid crowding local flockmates

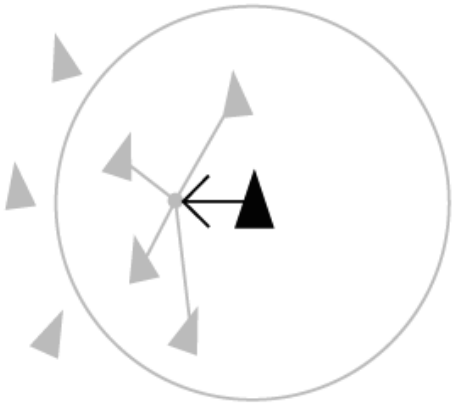
Alignment:
Steer toward the average heading of local flockmates

Cohesion:
Steer to move toward the average position of local flockmates

```
PROCEDURE move_all_to_new_positions()

    Vector v1, v2, v3, v4, …
    Boid b

    FOR EACH BOID b
        v1 = rule1(b)
        v2 = rule2(b)
        v3 = rule3(b)
        v4 = rule4(b);

        b.velocity = b.velocity + v1 + v2 + v3 + v4 + …
        b.position = b.position + b.velocity
    END

END PROCEDURE
```

Rule 1: Move towards the centre

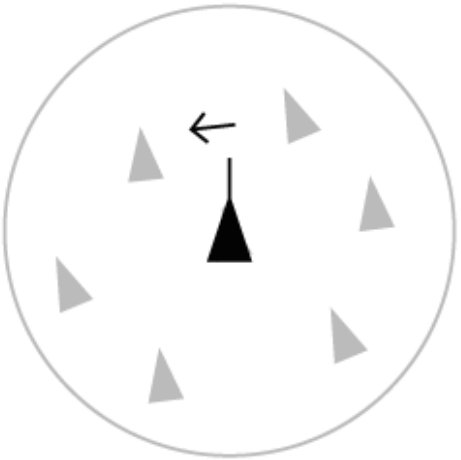
```
PROCEDURE rule1(boid b_J)

    Vector pc_J

    FOR EACH BOID b
        IF b != b_J THEN
            pc_J = pc_J + b.position
        END IF
    END

    pc_J = pc_J / N-1

    RETURN (pc_J - b_J.position) / 100

END PROCEDURE
```



Cohesion:
Steer to move toward the average position of local flockmates

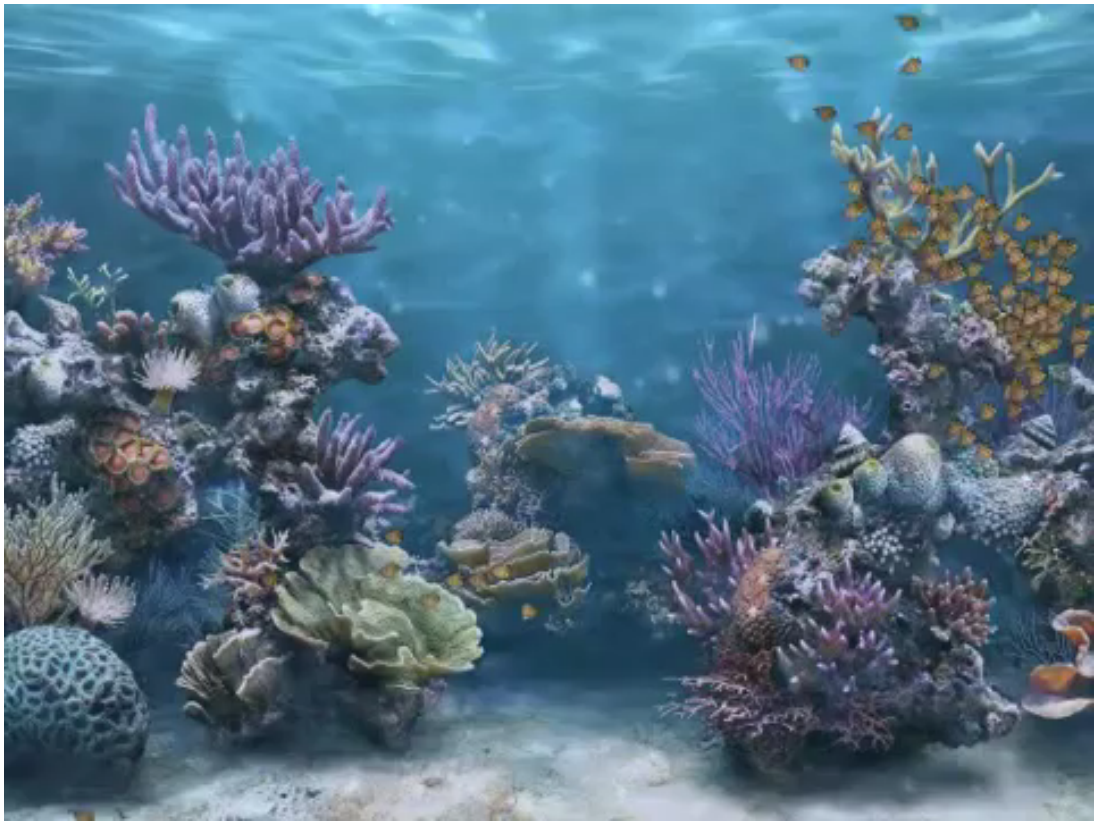
Rule 2: Steer towards average heading

```
PROCEDURE rule2(boid b_J)

    Vector pv_J

    FOR EACH BOID b
        IF b != b_J THEN
            pv_J = pv_J + b.velocity
        END IF
    END

    pv_J = pv_J / N-1

    RETURN (pv_J - b_J.velocity) / 8

END PROCEDURE
```



Alignment:
Steer toward the average heading of local flockmates

Simulation: Boid's Algorithm



Murmuring birds

Example Tool

- An example tool is available here:
  - http://www.red3d.com/cwr/boids/
  - http://harry.me/blog/2011/02/17/neat-algorithms-flocking/
  - http://anoopelias.github.io/boids/#

Emergent Behavior

- Emergent Behaviors can be a very powerful tool in simulations, virtual worlds and games.
- What seems to be complex does not actually have to be complex.
    - Illusion of Intelligence
- In some cases simulations with less input have proven to be more accurate than systems that takes all input into account.
- It can be difficult to judge the impact an input has on the final result.
    - This is the case in much bad science. What is causing what (causation)?

Summary

- An agent is an entity that perceives its environment and acts upon it.
- Single agent and multiagent systems have many practical applications.
- To develop the agent program the developers need to have a good knowledge in AI:
    - Planning
    - Search/pathfinding
    - Learning
    - Language processing
    - …

That was all for this lecture

Poll time: http://etc.ch/C5mX

Acknowledgements

Dr. Johan Hagelbäck

Linnæus University

johan.hagelback@lnu.se

http://aiguy.org