Real-world Multi-agent Systems

Introduction

Akin Delibasi
Lecturer in Distributed Systems
Computer Science
University College London, UK

MAS course lecturer

Akin Delibasi

Lecturer

Department of Computer Science

a.delibasi@ucl.ac.uk

University College London, UK

Assessment

- 50 marks, Examination: after week 43 (UCL academic calendar), starting from week 44 (22 June 2025 onwards).
- 50 marks, Report, submission at week 16. 12 December 2024
  - 15 marks, Clarity: Clear structure with a flowing, logical argument.
  - 10 marks, Robot Test Simulation
  - 10 marks, Real World Robot Application Task 1 Completion
  - 10 marks, Real World Robot Application Task 2 Completion
  - 5 marks, Additional Creativity

UCL

UCL Academic Calendar

AUGUST

WK	M	T	W	T	F	S	S
1	26	27	28	29	30	31	

SEPTEMBER

WK	M	T	W	T	F	S	S
1							1
2	2	3	4	5	6	7	8
3	9	10	11	12	13	14	15
4	16	17	18	19	20	21	22
5	23	24	25	26	27	28	29
6	30						

OCTOBER

WK	M	T	W	T	F	S	S
6		1	2	3	4	5	6
7	7	8	9	10	11	12	13
8	14	15	16	17	18	19	20
9	21	22	23	24	25	26	27
10	28	29	30	31			

NOVEMBER

WK	M	T	W	T	F	S	S
10					1	2	3
11	4	5	6	7	8	9	10
12	11	12	13	14	15	16	17
13	18	19	20	21	22	23	24
14	25	26	27	28	29	30	

DECEMBER

WK	M	T	W	T	F	S	S
14							1
15	2	3	4	5	6	7	8
16	9	10	11	12	13	14	15
17	16	17	18	19	20	21	22
18	23	24	25	26	27	28	29
19	30	31					

JANUARY

WK	M	T	W	T	F	S	S
19			1	2	3	4	5
20	6	7	8	9	10	11	12
21	13	14	15	16	17	18	19
22	20	21	22	23	24	25	26
23	27	28	29	30	31		

FEBRUARY

WK	M	T	W	T	F	S	S
23						1	2
24	3	4	5	6	7	8	9
25	10	11	12	13	14	15	16
26	17	18	19	20	21	22	23
27	24	25	26	27	28		

MARCH

WK	M	T	W	T	F	S	S
27						1	2
28	3	4	5	6	7	8	9
29	10	11	12	13	14	15	16
30	17	18	19	20	21	22	23
31	24	25	26	27	28	29	30
32	31						

APRIL

WK	M	T	W	T	F	S	S
32		1	2	3	4	5	6
33	7	8	9	10	11	12	13
34	14	15	16	17	18	19	20
35	21	22	23	24	25	26	27
36	28	29	30				

MAY

WK	M	T	W	T	F	S	S
36				1	2	3	4
37	5	6	7	8	9	10	11
38	12	13	14	15	16	17	18
39	19	20	21	22	23	24	25
40	26	27	28	29	30	31	

JUNE

WK	M	T	W	T	F	S	S
40							1
41	2	3	4	5	6	7	8
42	9	10	11	12	13	14	15
43	16	17	18	19	20	21	22
44	23	24	25	26	27	28	29
45	30						

JULY

WK	M	T	W	T	F	S	S
45		1	2	3	4	5	6
46	7	8	9	10	11	12	13
47	14	15	16	17	18	19	20
48	21	22	23	24	25	26	27
49	28	29	30	31			

AUGUST

WK	M	T	W	T	F	S	S
49					1	2	3
50	4	5	6	7	8	9	10
51	11	12	13	14	15	16	17
52	18	19	20	21	22	23	24

Lectures

1. Introduction of multi-agent systems
2. Sensing and Perception
3. Control of agents
4. Motion planning
5. Advanced optimization

   Reading week (Nov 4 5 6 7 8)

6. Multi-agent planning
7. Machine learning and AI
8. Multi-agent learning
9. Guest lecture, real-world case studies
10. Revision lecture & QA

Lab Practical

Week 0: Induction

Week 1: Forming teams, each team 3 people; gitclone and install, set up codebase.

Week 2: Cary out the unit tests of hardware functionalities

Week 3: Set up pybullet simulation of turtlebots - Simple perception tasks (LiDAR, ArUco marker)

Week 4: Test basic motion planning of one turtlebot in simulation, library provided as part of codebase

Week 5: Single-robot planning/navigation in simulation (Each student).

Reading week (Week 6, week 11 in academic calendar). Lab will still be open. Set up mockup arena using building blocks; Set up external cameras.

Week 7: Perform real single-robot planning/navigation (Each student).

Week 8: Multi-robot planning in simulation (Each student). Task 1 completion: search and find the target, eg a red target block in one of the rooms. Task 2 completion: reach (fetch) and return.

Week 9: Multi-robot planning in real arena (Each student). Task 1 completion

Week 10: Multi-robot planning in real arena (Each student). Task 2 completion

Week 11: Multi-robot planning in real arena

You will learn

Practical skills and tools:

1. Designing and implementing autonomous control and planning of turtle bots.
2. Applying control and signal processing of real robots (turtle bots).
3. Signal processing skills on real robots.
4. Experience with wireless communication, protocols, data transmission.
5. Developing distributed planning algorithms for multi-agent systems.
6. *Utilizing LLMs for decision-making processes.
7. Exploring real-world applications of multi-agent systems, such as intelligent logistics and transportation (lab sessions).
8. Hands-on experience in designing and implementing real multi-agent systems.

Report – 50 marks

- **Report writing quality: 15%**
- **Simulation & setup: 10%** (Multi-bot task 1; task 2)
- **Robot experimental results: 10%** (Multi-bot task 1)
- **Robot experimental results: 10%** (Multi-bot task 2)

- **Innovation: 5%**

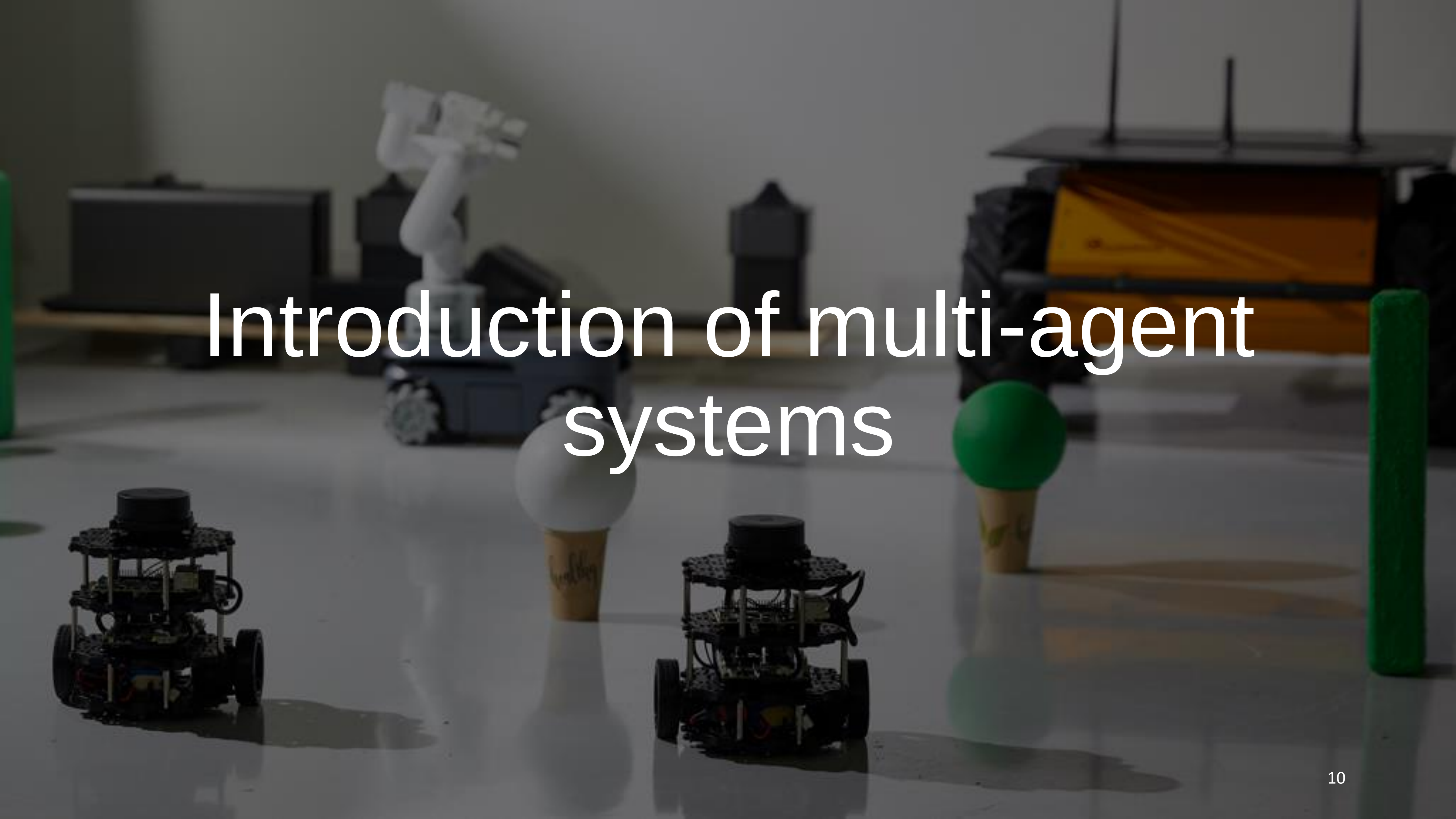
DECEMBER

WK	M	T	W	T	F	S	S
14							1
15	2	3	4	5	6	7	8
16	9	10	11	12	13	14	15
17	16	17	18	19	20	21	22
18	23	24	25	26	27	28	29
19	30	31					

Report – 50 marks

The report should include the sections below:

- **Introduction**: A general introduction of robot planning and multi-robot planning.
- **Methodology or Methods**: Explain any related techniques, such as signal processing, image recognition, and specific planning algorithms that are used in the simulation and experiments.
- **Results Section**: It will include two parts. Part 1 is the result of simulation, and Part 2 is the result of real robot experiments. In the results section, you need to include <u>data analysis and data plots</u> for both simulation and the real experimental results (please present single-robot and multi-robot separately).
- **Conclusion Section**: Summarize the main findings from both simulation and real robot experiments; Highlight what the successful achievements are; Discuss the limitations, unsolved problems, future work, potential real-world applications of the research.
- **Acknowledgments**: Thank group members and supporting individuals.
- **References**: List all cited references.



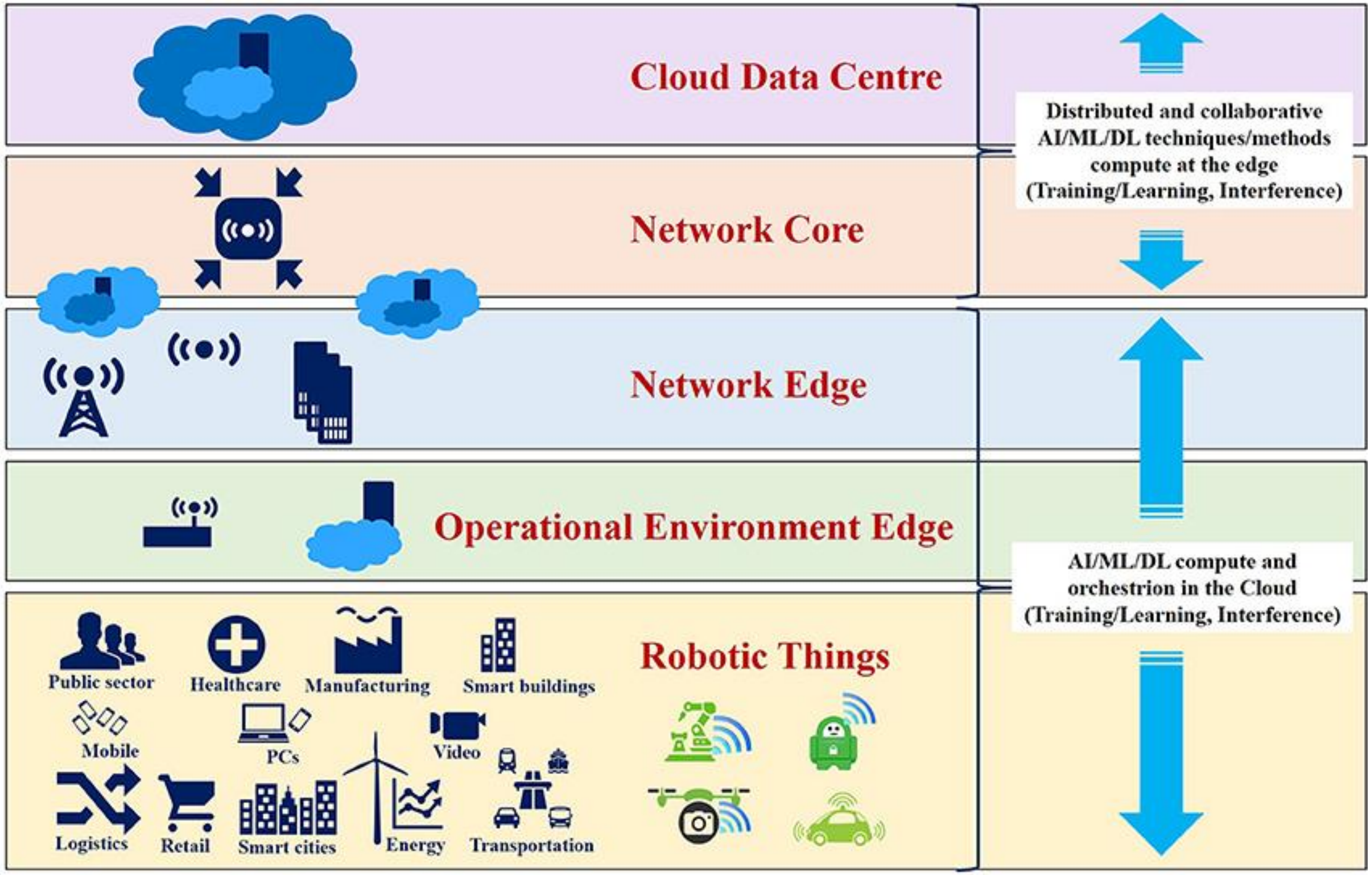
Introduction of multi-agent systems

Introduction

- MAS in the context of system engineering for internet of things
- Definition of multi-agent systems
- Core concepts of multi-agent systems
- Reasoning and decision-making

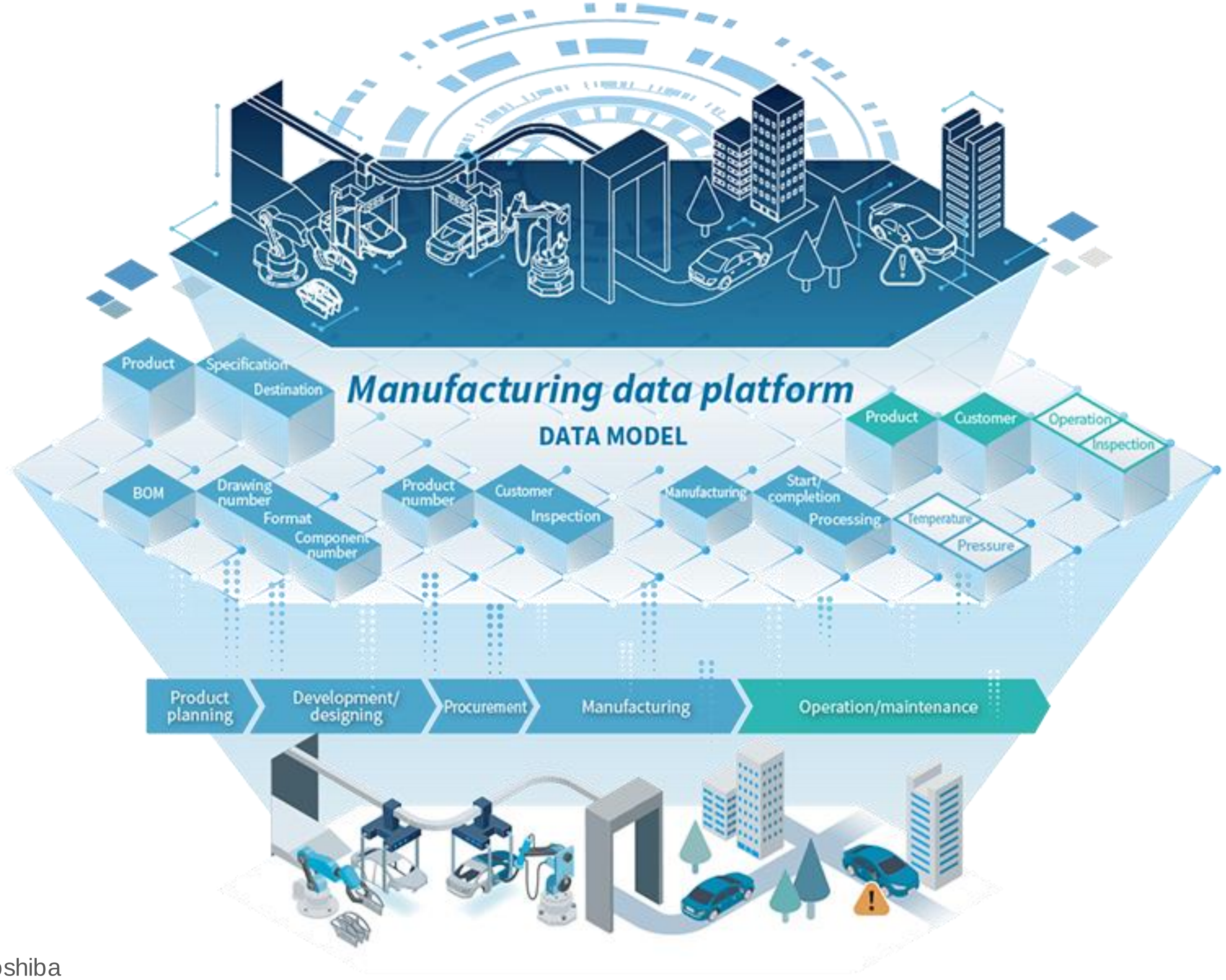
System engineering for internet of things

- Interdisciplinary branch of engineering and science.



Internet of Robotic Things Intelligent Connectivity and Platforms, https://doi.org/10.3389/frobt.2020.00104

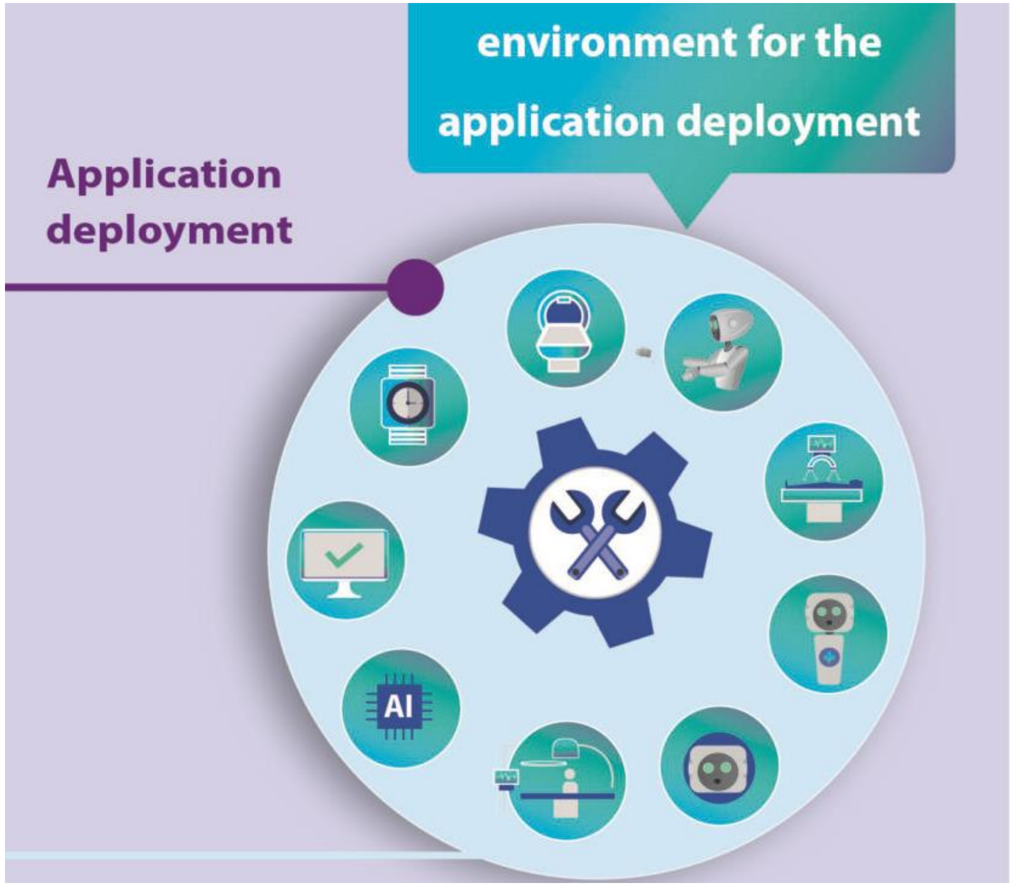
System engineering for internet of things



www.global.toshiba

Multi-agent systems

- Multi-agent systems has a wide range of fields and applications, including:
  - ❑ Artificial intelligence and cognitive systems
  - ❑ Robotics and unmanned/autonomous systems
  - ❑ Distributed systems and networks
  - ❑ Social systems and social simulation
  - ❑ Economics and game theory
  - ❑ Control systems and consensus algorithms
  - ❑ Transportation and logistics
  - ❑ Power systems and smart grids
  - ❑ Finance and e-commerce



Past and now

- The study of artificial intelligence (AI) and decentralised systems is at the heart of the discipline of multi-agent systems.
- Initially, the focus of AI research was mostly on constructing intelligent machines capable of doing certain tasks, such as playing chess or solving mathematical problems.

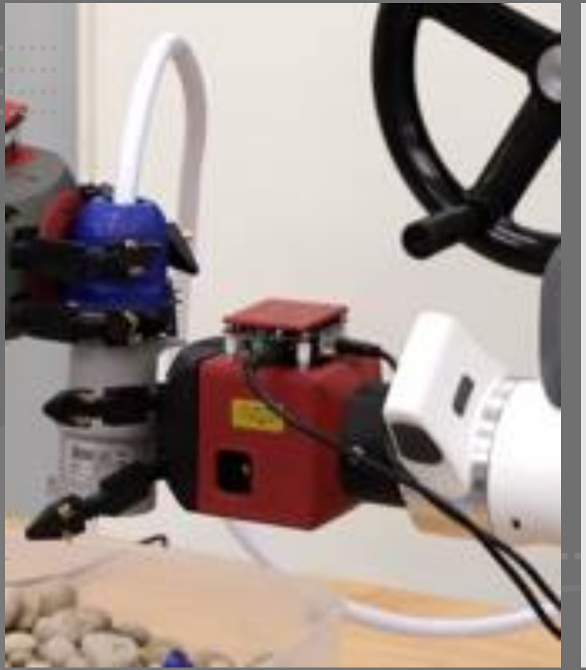
Past and now

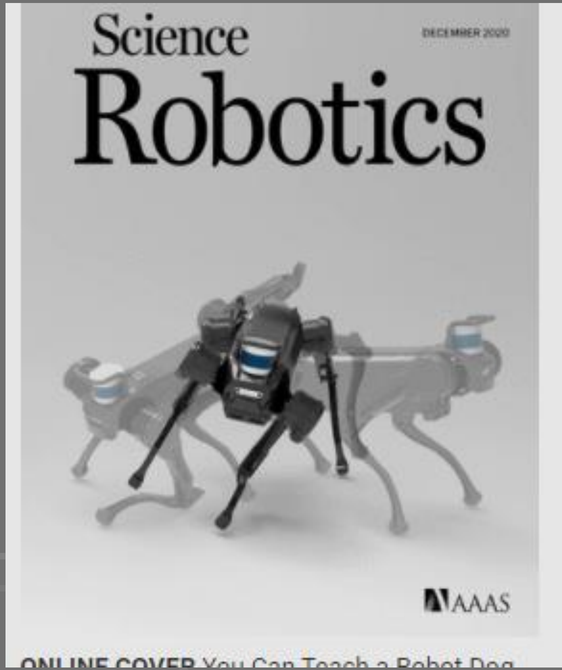
- In 1980s-90s, researchers began to explore the idea of creating collections of intelligent agents that could work together to accomplish more complex tasks.

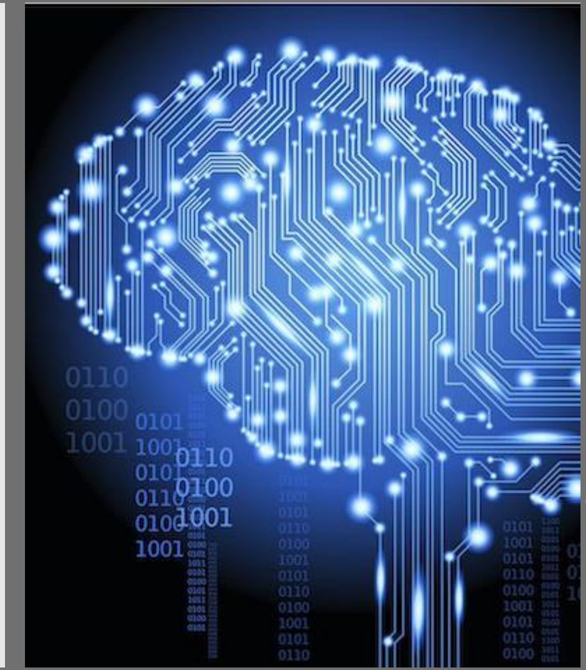
This led to the development of the field of multi-agent systems, which studies the behavior of groups of agents and their interactions.

- In the early 2000s, the field of multi-agent systems began to expand rapidly on various aspects, such as:
  - Communication and coordination
  - Game theory and decision making
  - Learning and adaptation
  - Control and coordination
  - Interaction with humans

UCL







- Multi-expert learning of adaptive legged locomotion
- Paper: https://www.science.org/doi/10.1126/scirobotics.abb2174

- Swarm of micro flying robots in the wild
- Paper: https://www.science.org/doi/10.1126/scirobotics.abm5954

New era, new definition

- Recent developments in machine learning and artificial intelligence, as well as increased availability of sensor data and computer resources, have pushed the limits and boundaries of **multi-agent systems** even farther.

- Today, multi-agent systems are being applied in a wide range of fields: from **robotics and unmanned systems** to social systems and social simulation; from **transportation and logistics** to finance and economics.

- Cutting edge of MAS: develop more **advanced** agents, with features such as self-awareness, and in exploring new application areas such as multi-robots, and the Internet of Things (IoT).

Scope & focus

Robotics and autonomous systems: to control and coordinate the behavior of groups of robots and unmanned vehicles. Multi-robot systems to perform tasks that are beyond the capabilities of a single robot.

Applications

- **Transportation and logistics**: to optimize transportation and logistics systems, eg, fleet management, logistics planning.
- **Manufacturing, supply chain**: to optimize the production process, plan, and schedule the manufacturing process, and optimize the supply chain.
- **Healthcare**: to support healthcare systems, such as patient monitoring, clinical decision support, assistive devices, smart care homes.

Scope & focus

- We are *not* covering:
  - ❑ Power systems and smart grids
  - ❑ Finance and economics
  - ❑ Social systems and social simulation
  - ❑ Entertainment and Gaming
  - ❑ Recommendation marketing

The core technologies can be applied and transferred to some of the domains above.

Key concepts

What is an Agent?

Definition: An agent is an autonomous entity that operates within a multi-agent system. It possesses the ability to perceive its environment, make decisions, and take actions to achieve its goals.

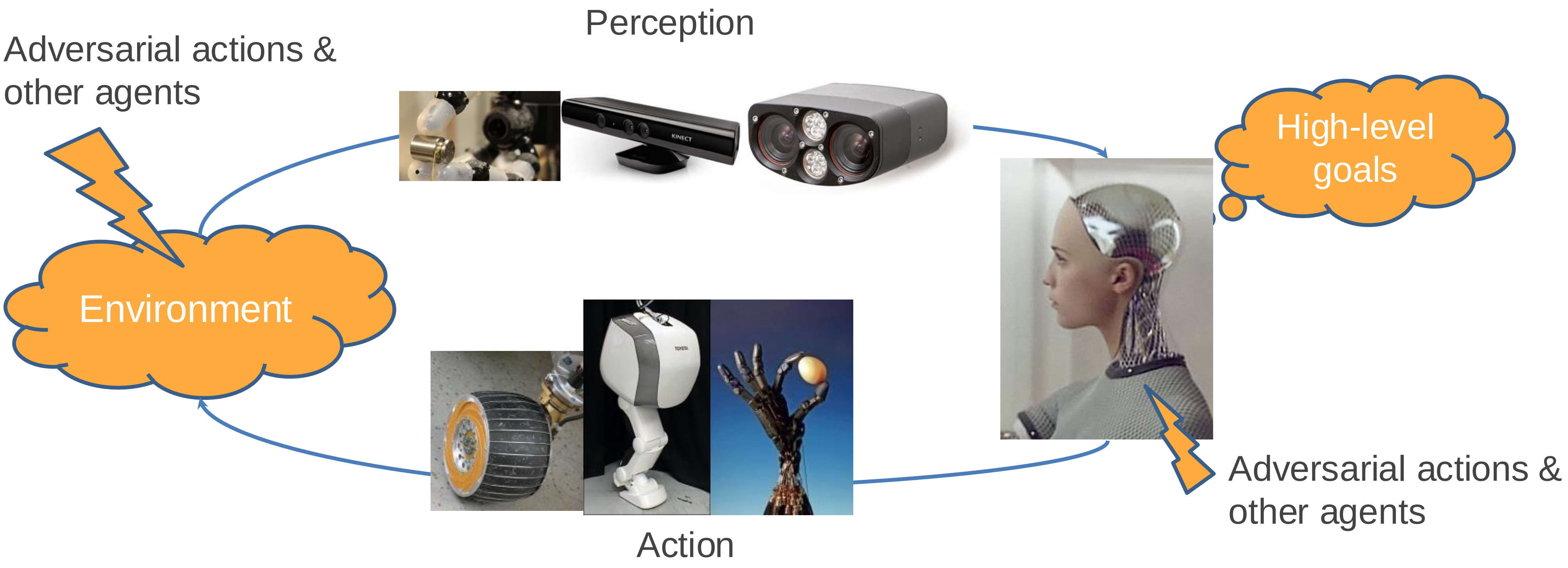
Definition

Multi-agent systems

- A multi-agent system (MAS) is multiple **autonomous (decision-making) agents** that interact with one another and with their environment in order to achieve a shared/conflicting goal(s) or set of goals.
- Agents in a MAS can have varying skills, goals, and decision-making processes, and they can operate in a centralized or decentralized/distributed manner.
- Autonomous agents act by itself, given high-level goals (from user or another AI).
- Ability to cooperate, coordinate, and negotiate with each other.

Intelligent Agents

- Problem: How to take actions, achieve high-level goals, using limited perception and incomplete knowledge of environment and other agents?



Intelligent Agents

- An intelligent agent (IA) is a software/hardware system that is able to perform tasks **autonomously** and make decisions based on its environment and goals.
- Intelligent agents are programmed to act rationally: take actions of <u>high probability</u> to achieve their goals, and can adapt behavior based on new information or changes in the environment.
- They can also see and reason about their surroundings.

Characteristics of an Agent

- **Autonomy**: Agents act independently, making decisions without external control.
- **Sensing**: They can sense and perceive information from their environment.
- **Reasoning**: Agents have the capability to reason and make decisions based on available data.
- **Communication**: They can communicate with other agents to share information or coordinate actions.
- **Proactiveness**: Agents can initiate actions to achieve their goals.
- **Reactivity**: They respond to changes in their environment or other agents' actions.

Types of Agents

Agents in multi-agent systems:

- **Software Agents (eg, LLMs)**: purely software-based agents in digital realm.
- **Robotic Agents**: Physical robots equipped with sensors and actuators that interact with the physical world.
- Human Agents: Humans can also be considered as agents when participating in multi-agent systems.
- **Hybrid Agents**: agents combined with software and hardware.

Intelligent Agents

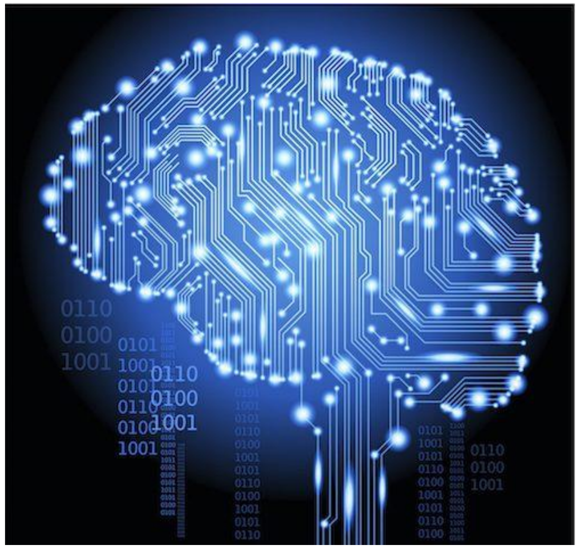
Examples of (single) intelligent agents/systems:

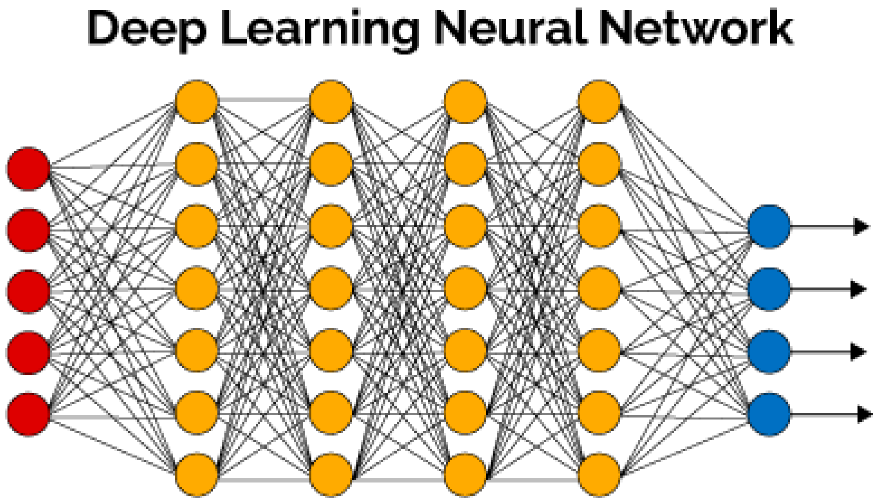
- A self-driving car senses its environment and makes decisions about when to brake or turn
- Online assistant that can schedule appointments and answer questions
- A recommendation system suggests products based on users' browsing and purchase history.

Intelligent agents can be simple or complex, and they can operate in a wide range of environments, from a single device to a distributed network of systems.

Intelligent Agents – the mind

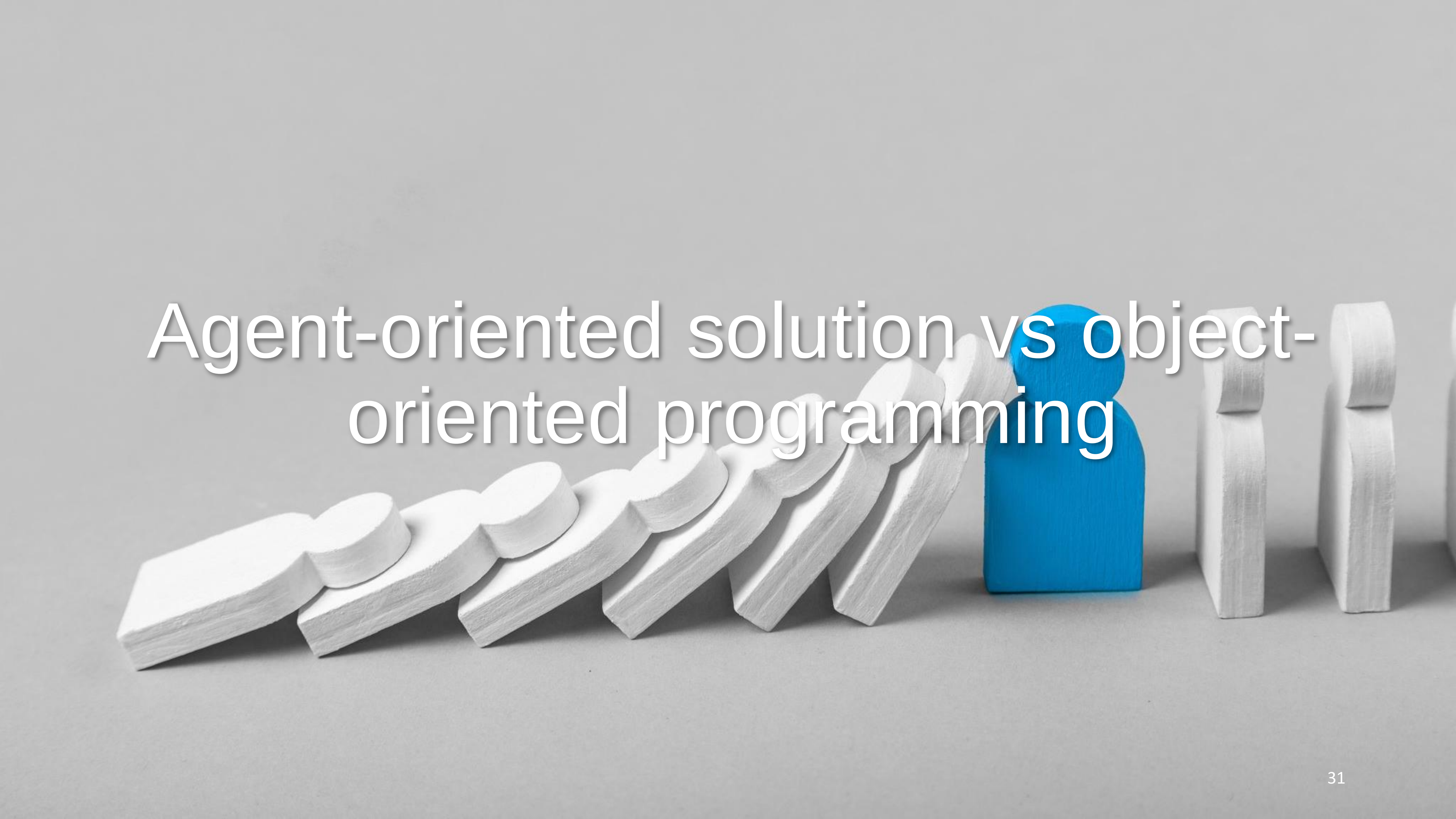
- Intelligent agents, autonomous systems/robots.
- In machine learning, an ***agent*** is often referred to an AI system, typically neural networks.
- It is more about software.





Intelligent Agents – the body

- Intelligent agents, autonomous systems/robots.
- In robotics, an agent is a physical robot, or an autonomous system.



Agent-oriented solution vs object-oriented programming

Agent-oriented solution

- What are the differences between an agent or agent-oriented solution vs other software paradigms, such as object-oriented programming (OOP)?

- **Agent-Oriented**: Agents in multi-agent systems are typically designed to have independent decision-making. They have their goals, beliefs, and reasoning capabilities. This autonomy allows agents to act independently and produce complex behaviors based on internal state and interactions with other agents.

- **OOP**: objects encapsulate data and behavior, but lack the autonomy and independent decision-making. Objects respond to method calls from external sources but don't inherently possess goals or beliefs.

UCL

Agent-oriented solution

- Goal-Directed vs. Object-Centered
- **Agent-Oriented**: Agents are goal-directed. Agents use reasoning abilities to formulate plans and take actions to accomplish these goals. They negotiate, cooperate with other agents to achieve objectives.

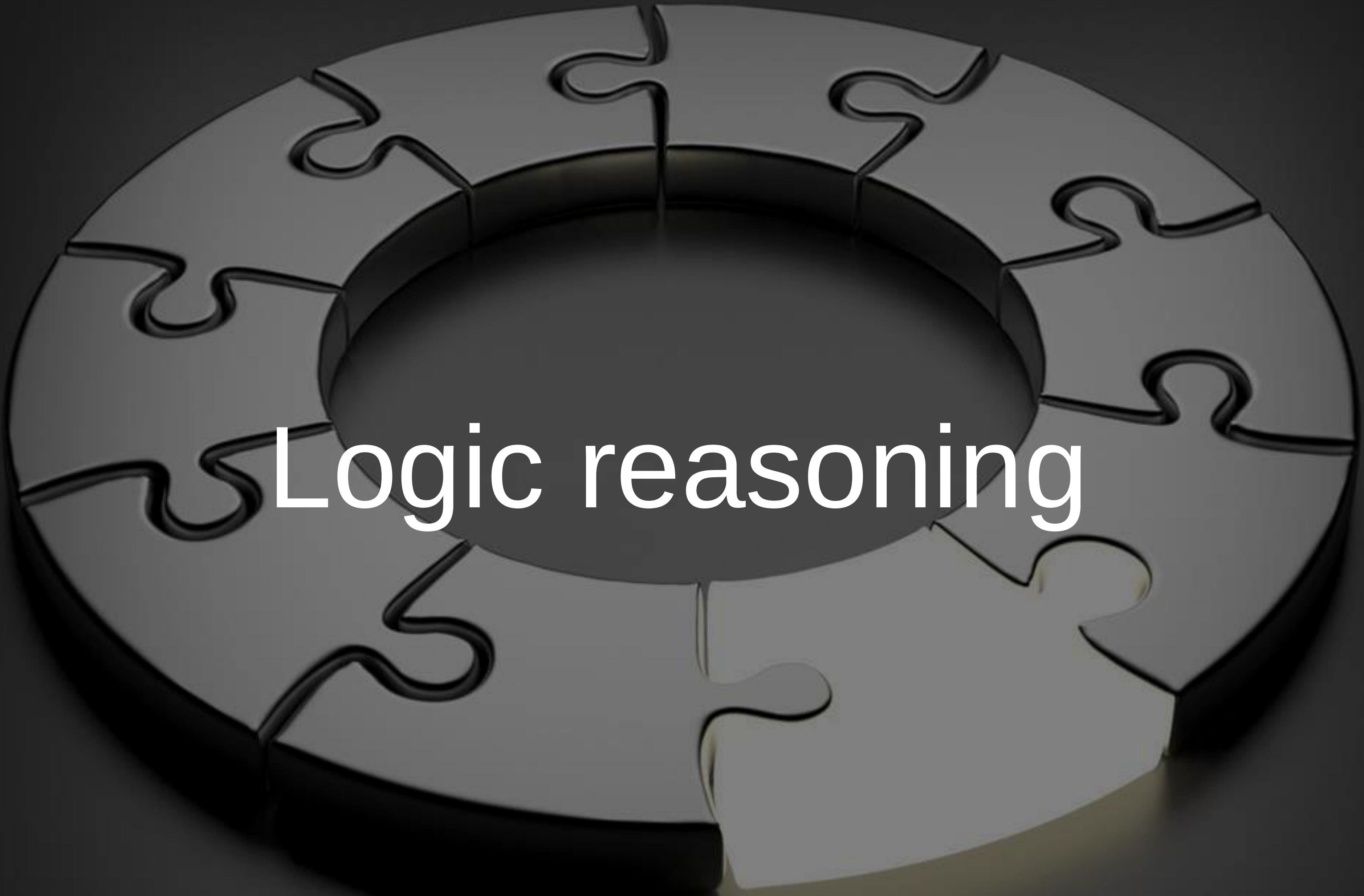
  => Emergent Behavior

- **OOP**: In OOP, objects represent entities with attributes (data) and methods (actions). These objects are instances of classes. The data and behavior associated with a particular object are encapsulated within its respective class.

  => Structured Systems

Agent-oriented solution

- Dynamic interaction vs. Statically defined interaction
- **Agent-Oriented**: Multi-agent systems are well-suited for **dynamic and uncertain** environments. Agents adapt to changing conditions, make decisions in real-time, and respond to unforeseen events. Agent-oriented solutions are flexible and capable of handling environments with high variability.
- **OOP**: Object-oriented systems tend to have **fixed structures** and may not be as adaptable to dynamic environments without significant modifications.



Logic reasoning

Boolean logic operations

<table>
<tr><th>AND (Conjunction)</th><th>OR (Disjunction)</th><th>XOR (Exclusive OR)</th><th>NOT (Negation)</th></tr>
<tr><td>x && y, and</td><td>x || y, or</td><td>x ^ y, or !=</td><td>not, !</td></tr>
</table>

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

A	B	$A \veebar B$
0	0	0
0	1	1
1	0	1
1	1	0

A	$\neg A$
0	1
1	0

Python code example

main.py

Save Run

```python
# Define boolean variables
A = True
B = False

# Perform logic operations
result_and = A and B
result_or = A or B
result_xor = A != B
result_not = not A

# Print results
print(f'A AND B: {result_and}')
print(f'A OR B: {result_or}')
print(f'A XOR B: {result_xor}')
print(f'NOT A: {result_not}')
```

Shell

```
A AND B: False
A OR B: True
A XOR B: True
NOT A: False
>
```

Deductive Reasoning

- A fundamental form of logical reasoning, a formal process of drawing deductive **inferences**. An inference is deductively valid if its conclusion follows logically from its **premises**.

- Deductive reasoning: you start with a general rule or principle, and apply it to a specific case or situation through inferences.

Examples: Deductive logic arguments

Premise	All insects have exactly six legs.	All insects have exactly six legs.
Premise	Spiders have eight legs.	All ants are insects.
Conclusion	Therefore, spiders are not insects.	Therefore, all ants have exactly six legs.

Deductive Reasoning

- Modus Ponens: If "A implies B" and "A" is true, then "B" must be true.
  - Premises: $A \rightarrow B, A$
  - Conclusion: B

- Modus Tollens: If "A implies B" and "B" is false, then "A" must be false.
  - Premises: $A \rightarrow B, \neg B$
  - Conclusion: $\neg A$

- Hypothetical Syllogism: If "A implies B" and "B implies C," then "A implies C.“
  - Premises: $A \rightarrow B, B \rightarrow C$
  - Conclusion: $A \rightarrow C$

Deductive Reasoning

- Disjunctive Syllogism: If "A or B" is true, and "A" is false, then "B" must be true.
  - Premises: $A \lor B, \neg A$
  - Conclusion: B
- Constructive Dilemma: If "A implies B" and "C implies D," and "A or C" is true, then "B or D" must be true.
  - Premises: $A \rightarrow B, C \rightarrow D, A \lor C$
  - Conclusion: $B \lor D$
- Destructive Dilemma: If "A implies B" and "C implies D," and "Not B or Not D" is true, then "Not A or Not C" must be true.
  - Premises: $A \rightarrow B, C \rightarrow D, \neg B \lor \neg D$
  - Conclusion: $\neg A \lor \neg C$

UCL

Case study

- Context: multi-robot cooperation in warehouse
- Robots: different robots have different capabilities, like lifting heavy objects (L), transporting fragile goods (T), or moving pallets (P).
- Task: each task requires certain abilities.
- Our goal:
  - ❑ We need to select the right robots for each task. Additionally, some tasks might need more than one robot working together.
  - ❑ Program a reasoning process to determine the best robot or combination of robots for each task.

Case study

```python
# Import itertools
import itertools

# Define robots and their capabilities
robots = {
    "Forklift": ["L", "P"],
    "Drone": ["T"],
    "Conveyor": ["P"],
}

# Define tasks
tasks = {
    "Task 1": ["L"],
    "Task 2": ["T"],
    "Task 3": ["P"],
    "Task 4": ["L", "T"],  # Requires both lifting and transporting
}
```

- Four tasks to be performed, like Task 1, Task 2, Task 3, and Task 4. Each task has specific requirements or abilities needed to complete it. For instance, Task 1 requires the ability to lift heavy objects (L), Task 2 needs the ability to transport fragile goods (T), and Task 3 requires the ability to move pallets (P).
- Task 4 is more complex because it requires both lifting (L) and transporting (T) abilities to complete.

Case study

```python
# Import itertools
import itertools
# Define robots and their capabilities
robots = {
    "Forklift": ["L", "P"],
    "Drone": ["T"],
    "Conveyor": ["P"],
}

# Define tasks
tasks = {
    "Task 1": ["L"],
    "Task 2": ["T"],
    "Task 3": ["P"],
    "Task 4": ["L", "T"],  # Requires both lifting and transporting
}
```

```python
# Deductive reasoning to assign tasks to robots (including cooperation)
task_assignments = {}

for task, requirements in tasks.items():
    suitable_robots = []

    # Check each robot's capabilities and match them to task requirements
    for robot, capabilities in robots.items():
        meets_requirements = all(req in capabilities for req in requirements)

        # Apply deductive reasoning
        if meets_requirements:
            suitable_robots.append(robot)
```

Case study

```python
    # Check cooperative robot assignments
    if not suitable_robots:
        for combo in itertools.combinations(robots.keys(), len(requirements)):
            combo_capabilities = [cap for robot in combo for cap in robots[robot]]
            meets_requirements = all(req in combo_capabilities for req in requirements)

            if meets_requirements:
                suitable_robots.extend(combo)
                break

    # Assign suitable robots to the task
    task_assignments[task] = suitable_robots
```

```python
# Display task assignments
for task, assigned_robots in task_assignments.items():
    if assigned_robots:
        print(f"{task} can be performed by: {', '.join(assigned_robots)}")
    else:
        print(f"No suitable robot found for {task}.")
```

```
Task 1 can be performed by: Forklift
Task 2 can be performed by: Drone
Task 3 can be performed by: Forklift, Conveyor
Task 4 can be performed by: Forklift, Drone
```

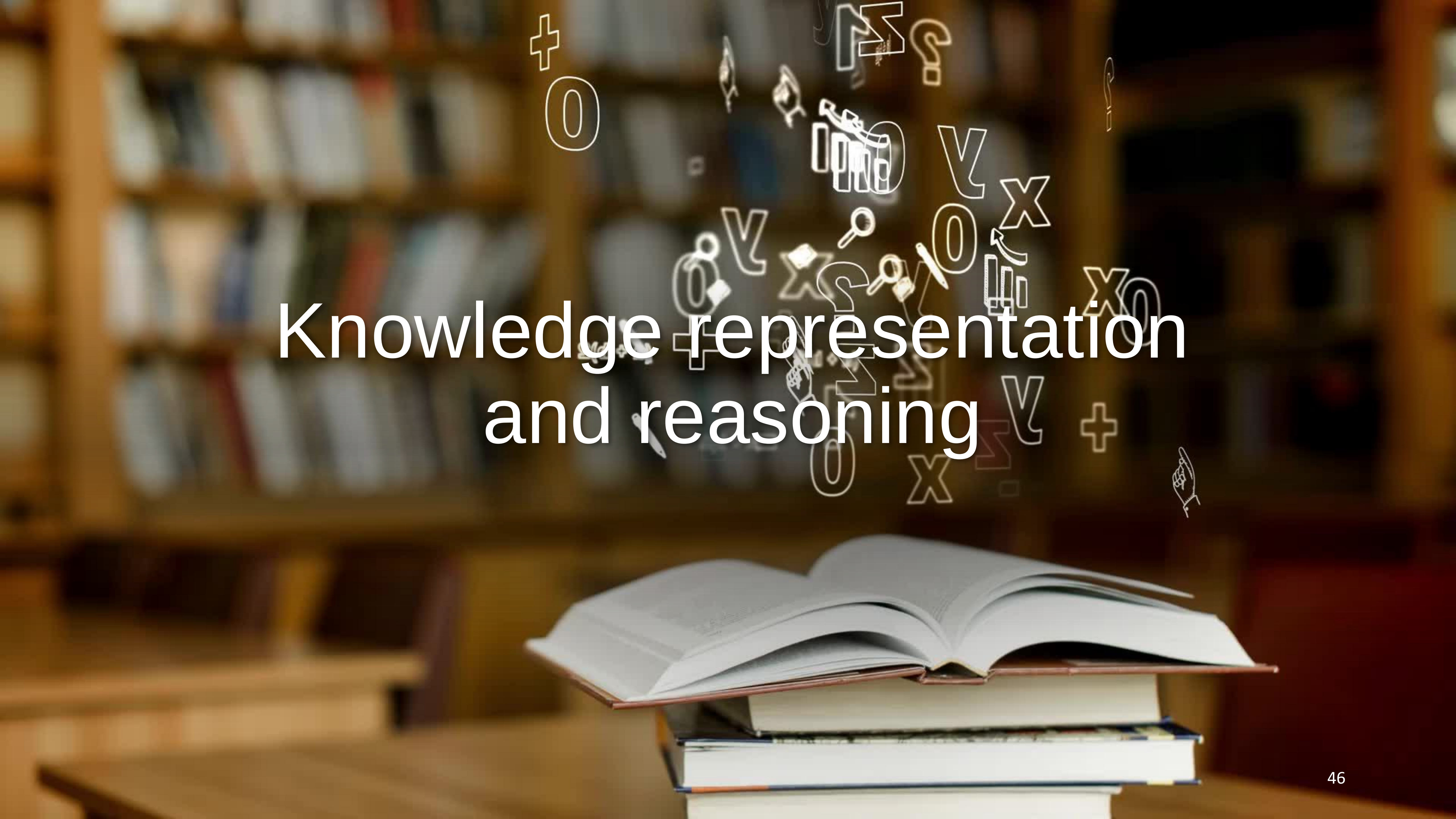
Case study

- In this code, the logic is straightforward and rule-based: searching and matching predefined symbols (capabilities vs task requirements) to make task assignments based on these straightforward matches.

- Using the same principle, we can have a database of capabilities from different robots, and a list of tasks with required capabilities, and use this code to find out robot-task matching pairs.

- Limitations: rules are explicit, and the representation of rules and knowledge needs to be represented by symbols. (Symbolic reasoning)

Knowledge representation and reasoning



Different Architectures

- Symbolic Reasoning Agents (1956-present):
  - These agents make decisions by manipulating symbols or abstract representations of knowledge.
  - The purest form of symbolic reasoning agents relies on explicit logical reasoning to decide what actions to take.
  - Symbolic reasoning agents use rules, logic, and knowledge representations to make decisions.

- Limitations: it relies on explicit rules and predefined symbols; requires manual work to symbolize the notions, knowledge, and rules.

Different Architectures

- Reactive Agents (1985-present):
  - Reactive agents represent a shift away from symbolic reasoning.
  - This movement emerged in response to the perceived limitations and complexities associated with purely symbolic reasoning.
  - Reactive agents focus on reacting directly to sensory input and the current state of the environment.
  - They prioritize real-time, immediate responses without relying heavily on symbolic knowledge representations.
  - Reactive agents often use simple, rule-based behaviors to respond to stimuli.

- Hybrid Agents (1990-present): combine the strengths of both reasoning-based and reactive architectures.

Common sense reasoning

- Preinstalled robot knowledge: When there's a dangerous situation or scene, the robot should not continue the task.
- Preinstalled robot capability dictionary:

  1. The robot can take action by itself given the existing motion library, such as move forward/backward, left, right, or plan obstacle free paths and navigate through an environment;

  2. if the robot doesn't know any actions to take, ask humans for help or send out help signals.
- Current situation: the robot's smoke detector detects smoke, and the temperature sensor has rising readings.
- Decisions to make: what the robot should do now?

?

Decisions by LLMs (GPT4)

Given the current situation where the robot's smoke detector detects smoke and the temperature sensor has rising readings, the robot should take the following actions:

1. **Stop any ongoing tasks**: As per the preinstalled knowledge, the robot should not continue its task in a dangerous situation.
2. **Move away from the source of heat and smoke**: Utilizing its existing motion library, the robot should move in a direction that takes it away from the increasing temperature and smoke.
3. **Send out a help signal or alert**: If the robot is unable to mitigate the situation by itself, it should ask for human help or send out help signals as per its preinstalled capabilities.

Remember, safety is always the top priority in such situations. The robot should ensure its own safety while also alerting others to the potential danger.

UCL

What are the limitations of traditional approaches?

Traditional AI methods and approaches are limited in decision-making in complex, real-world situations. Here are some of the key limitations:

1. **Lack of Adaptability**: Traditional AI methods are tasks specific and lack the ability to adapt to new situations or new unseen tasks.
2. **Limited Decision-Making**: Traditional AI systems follow pre-programmed rules and lack the ability to make informed decisions based on real-time data or changing situations.
3. **Inability to Handle Uncertainty**: Real-world situations involve uncertainties, which traditional AI methods struggle to handle. They typically require clear, structured input and may not perform well with incomplete or ambiguous information.
4. **Lack of Generalization and Common Sense**: Traditional AI systems often struggle to generalize their learning from one context to another.

New era of AI, new paradigm of MAS

Traditional Multi-Agent Systems

Traditionally, multi-agent systems are composed of individual agents programmed to follow predefined rules and logic. These agents interact with each other and their environment, typically in a **rule-based or behavior-driven** manner.

Fundamentally, before LLM, the **"Reasoning"** process is manually programmed, which have severe limitations:

- Limited Adaptability
- Complex Rule-Based Logic
- Lack of Natural Language Interaction

New era of AI, new paradigm

- Large Language Model (LLM), such as GPT-3, GPT-4, can be directly used as an **intelligent agent** (software) within a multi-agent system, which is a novel approach that can disrupt existing solutions.

- Fundamentally, before LLM, the **“Reasoning”** process is manually programmed, that was:
  - ❑ Not Generalisable
  - ❑ Not Scalable
  - ❑ No General Intelligence

Disruptive Role of LLMs

- LLMs, such as OpenAI's GPT series, bring disruptive capabilities to multi-agent systems:
- **Generic Intelligent Agents**: LLMs can serve as generic agents with common sense that adapt to new tasks by understanding and generating natural language instructions.
- **Inter-Agent Communication**: LLMs can be used by agents to communicate with each other, by translating text messages and instructions in natural language, overcoming barriers of standardizing protocols.
- **Reasoning and Contextual Understanding**: LLMs have the ability to reason over large amounts of text data and context. They can infer relationships, find relevant information, and make informed decisions based on textual input.

Disruptive Role of LLMs

- **Task Complexity**: LLMs can handle complex tasks that require reasoning over textual or domain-specific information. They can provide agents with valuable insights and decision
- **Adaptive Learning**: LLMs can be fine-tuned or adapted to specific tasks or environments using minimal labeled data. This adaptability allows them to learn new tasks quickly.

Knowledge representation and reasoning

Usefulness in our MAS

LLM can be used as a "Decision-Maker":

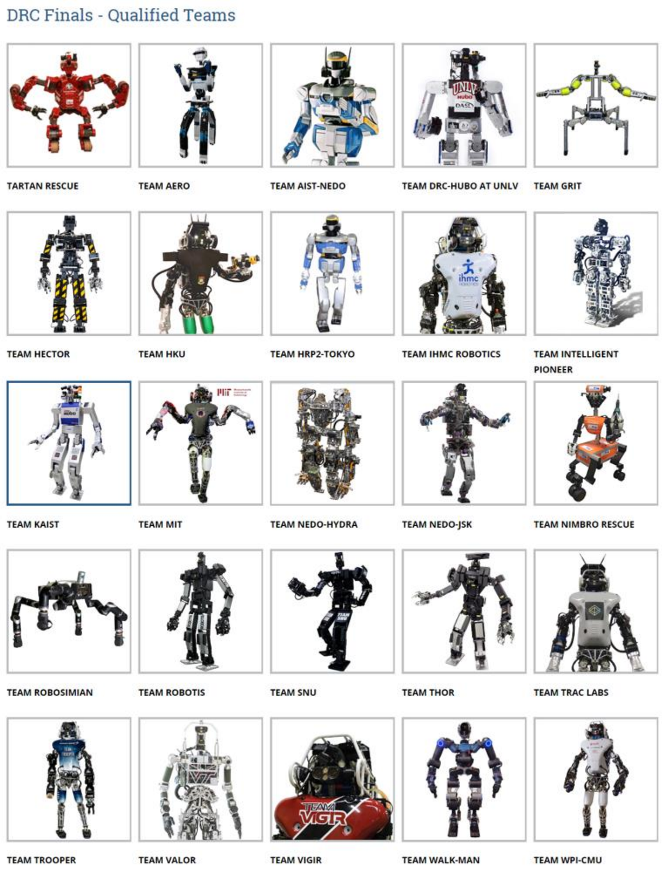
- Utilize the LLM to perceive sensor data and make navigation decisions.
- The LLM can understand natural language instructions and provide high-level navigation commands to the robot agent.
- LLM for seamless integration of perception, natural language interaction, and navigation control.

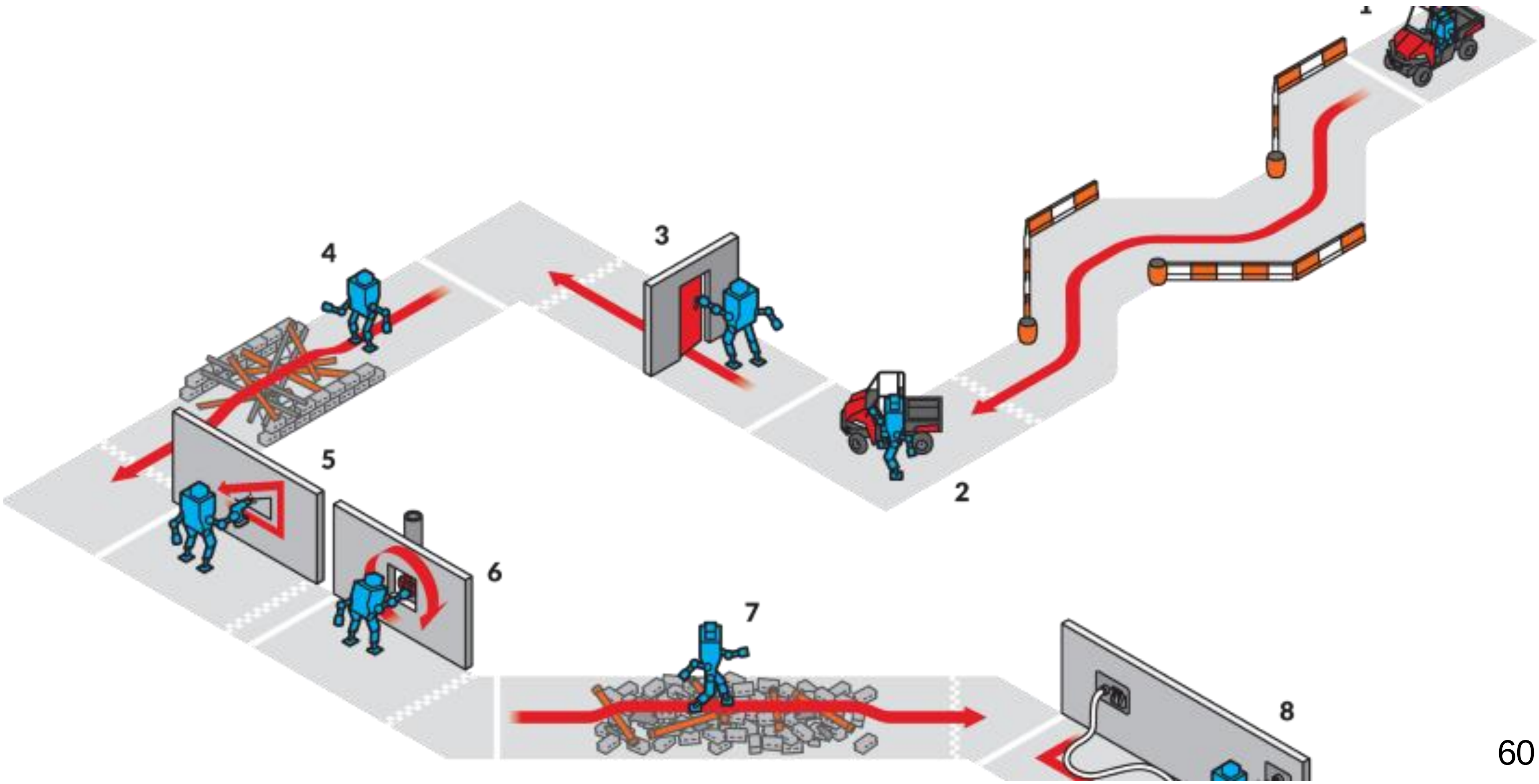
Usefulness in our MAS

1. **Dynamic Adaptation:** Multi-agent systems with LLMs can adapt dynamically to changing environments, tasks, or user requirements, reducing the need for manual reprogramming.
2. **Scalability:** LLM-based agents can scale to handle large numbers of agents and complex tasks, thanks to their ability to process and generate text at scale.
3. **Rapid Prototyping:** LLMs enable rapid prototyping and development of multi-agent systems, reducing the time and effort required to build intelligent agents.
4. **Truly intelligent behaviors**: LLMs are trained on massive data that captures common sense and exhibits an average adult level of reasoning.

Why intelligent behaviors matter?

Most robotic applications are particularly programmed for solving specific problems. However, what if we want **more universal or versatile machines**?



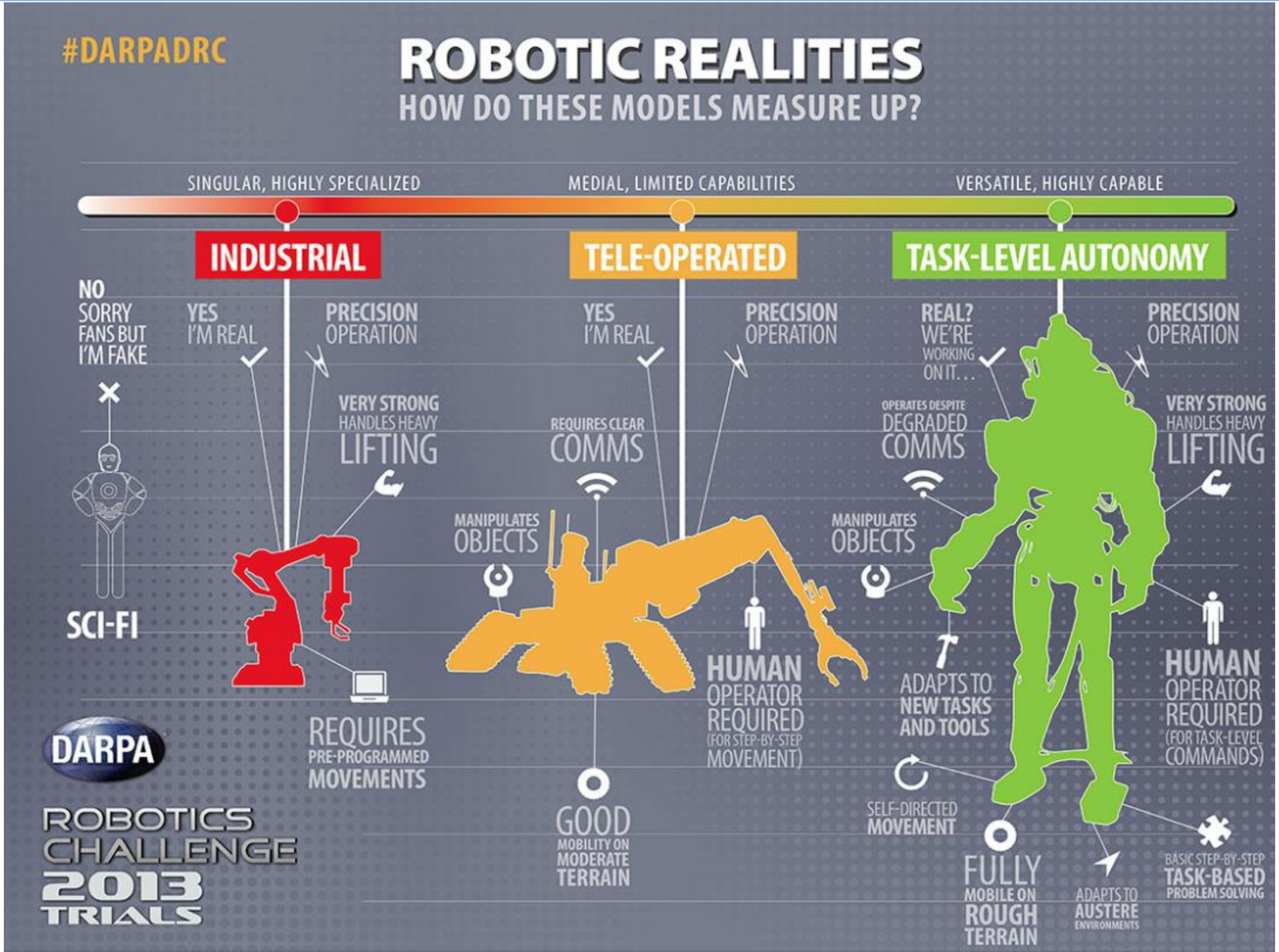


Why intelligent behaviors matter?

- In 2015, robot autonomy needed human intervention & supervision – human brains are behind the scene.

© University of Bonn, Autonomous Intelligent Systems

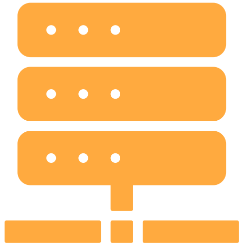
Roadmap for autonomous machines



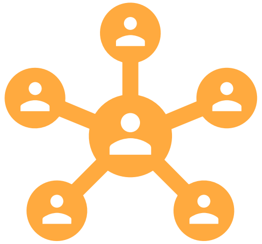
UCL

3 Pillars of MAS under the IoT

Individual intelligence – capabilities of agents themselves (distributed)



Communication, data and central intelligence (connectivity, data, computation)



Collective intelligence, interactions, and bahaviors (decentralized + centralized)