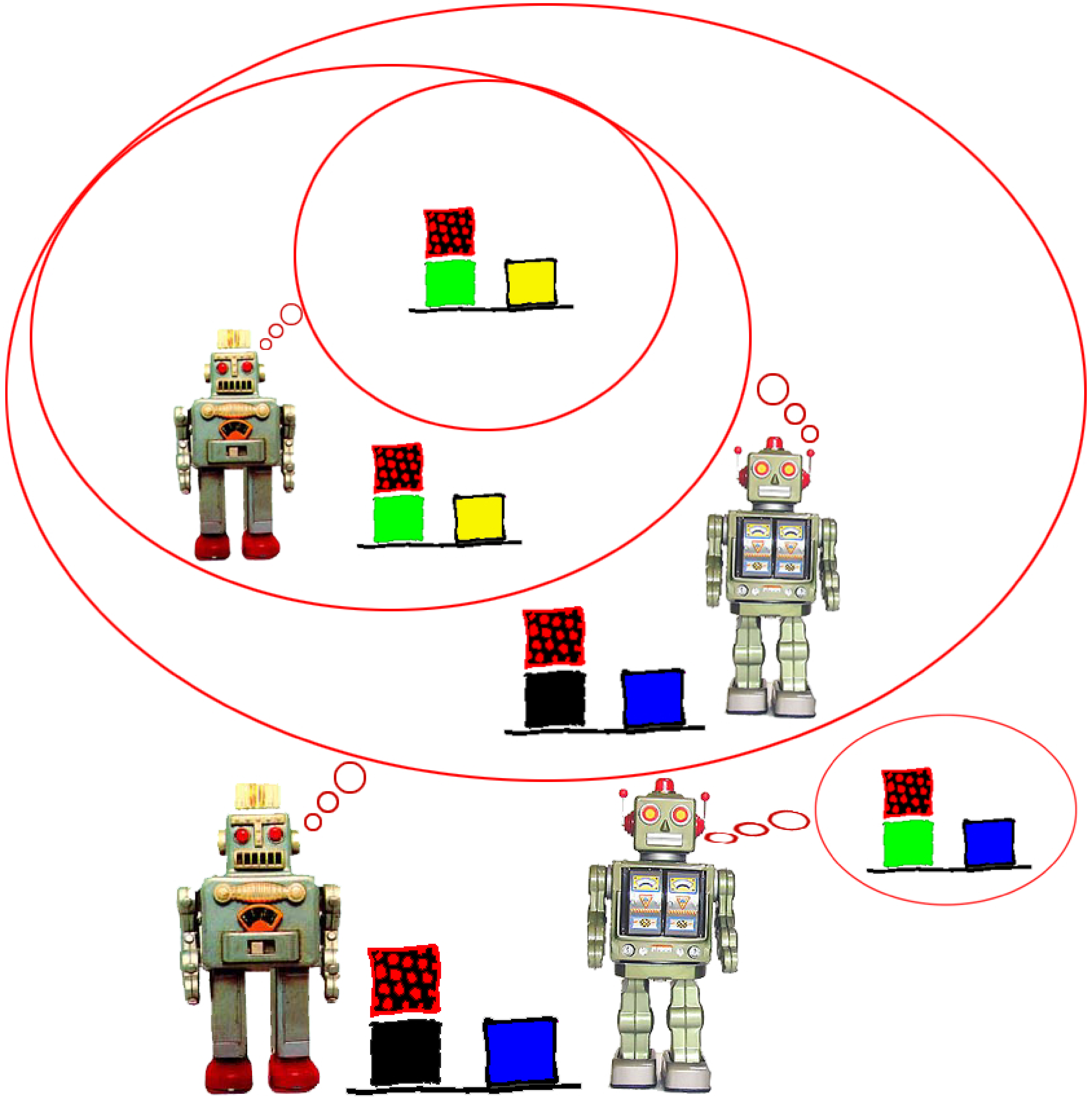
02285 AI and MAS

Week 1: Course introduction and AI search basics

About myself

Thomas Bolander

- Associate professor in **logic** and **AI** at **DTU Compute**.
- Current main **research topic**: To equip AI systems with a **Theory of Mind**.
- **Current teaching**:
  - 01017 Discrete Mathematics
  - 02180 Introduction to Artificial Intelligence
  - Artificial Intelligence at DIS.
  - 02285 AI and MAS.

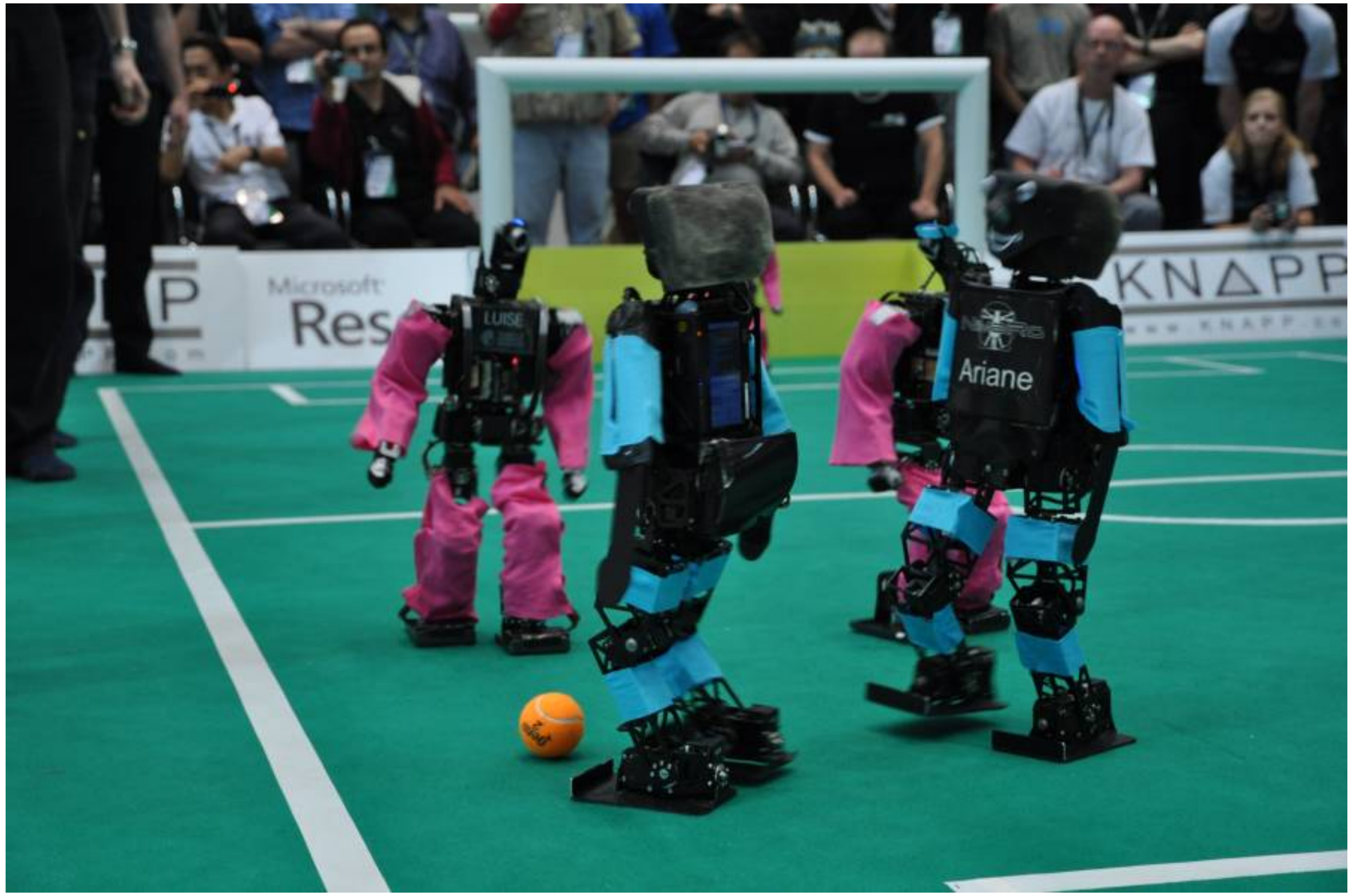


What is AI and MAS?

- **AI (Artificial Intelligence)** is “the science and engineering of making intelligent machines, especially intelligent computer programs” (John McCarthy).
- **Agent**: An autonomous entity which observes and acts upon an environment. It can e.g. be a computer program or a robot, but it can also be a human being. Usually a *goal directed* entity.
- **MAS (Multi-agent System)**: A system comprised of multiple interacting agents. These agents are usually required to be autonomous (independent) and to form a decentralised system (no controlling agent).

MAS examples

- Groups of **hospital robots** working in a shared environment.
- Agents for retrieving information on the **Web**.
- Teams of **soccer-playing robots**.



RoboCup 2009

General course information

The course consists of two parts of approximately equal length and workload:

- *First half:* **Lectures**, **exercise classes** and **2 mandatory group assignments** (2-3 students per group). Approximately 3.75 ECTS.
- *Second half:* **Mandatory programming project**. Approximately 3.75 ECTS. In groups of 3-5 students. Implementation (in programming language of your choice) and report. Concluded by a competition.

Demo of programming project

Assessment

The assessment is based exclusively on the written work, that is, the **3 mandatory assignments**. These are evaluated as a whole.

- **Assignment 1**: Groups of 2-3. Counts approximately 10%.
- **Assignment 2**: Groups of 2-3. Counts approximately 15%.
- **Assignment 3 (programming project)**: Groups of 3-5. Counts approximately 75%. Assessment based on group report and detailed group declaration (who did what).

Course material

- **Main textbook** is Russell & Norvig: *Artificial Intelligence—A Modern Approach*, 3ed, 2010.
- **Supplementary chapters on MAS** from other books (will be made available through file sharing).
- **Supplementary chapters on planning** from Geffner & Bonet: *A Concise Introduction to Models and Methods for Automated Planning*. Link to e-book from CampusNet welcome page.

About the Russell & Norvig textbook

Advantages:

- **The** standard textbook in AI.
- Covers very **broadly**: most major areas of AI.
- Rich in **examples**.
- Most people find it **inspiring** to read.

Disadvantages:

- Many details are **left implicit**, so things might seem deceptively simple. **Read carefully**.
- **Lack of technical details** and sometimes also **lack of mathematical precision** can make it more difficult to reach a deep understanding and be able to implement ideas and techniques.
- Some areas are **missing** (e.g. multi-agent systems) or **not completely up-to-date** (e.g. automated planning).

About the course curriculum

- The course deliberately prioritises **depth over breadth**.
- Major subjects covered: **automated planning** and **multi-agent systems**.
- Subjects are chosen according to **relevance for the programming project**. This means that everything covered in the course is relevant for the programming project, but not that everything covered can necessarily be *directly* applied in project.
- **Course goal**: To bridge academic AI research (often generic, domain-independent) with implementing a concrete AI system (domain-specific).
- **Trade-off** between generality and specificity in AI: computational efficiency, academic relevance, elegance, relevance for other applications, extendability, generalisability.

Other AI-relevant courses at DTU (and their relation to the Russell and Norvig textbook)

- **Advanced search methodologies** (Ch. 3-4 of R&N)
  - 02282 Algorithms for Massive Data Sets
  - 42137 Optimization Using Metaheuristics
- **Operations Research (OR)** (Ch. 4, 6 of R&N)
  - 42101 Introduction to Operations Research
  - 42114 Integer Programming
  - 42139 The Set Partitioning Optimization Model and its Application in Practical Scheduling Problems
  - 42142 Recent Research Results in Operations Research
- **Advanced logical methods in AI** (Ch. 7-9 in R&N)
  - 02156 Logical Systems and Logic Programming
  - 02281 Data Logic
  - 02287 Logical Theories for Uncertainty and Learning

Other AI-relevant courses at DTU

- **Knowledge-based reasoning** (Ch. 12 of R&N)
  - 02284 Knowledge-based Systems

- **Probabilistic reasoning and learning** (Ch. 13-21 of R&N)
  - 02417 Time Series Analysis
  - 02450 Introduction to Machine Learning and Data Modeling (general intro)
  - 02457 Non-Linear Signal Processing
  - 02460 Advanced Machine Learning
  - 02582 Computational Data Analysis
  - 02456 Deep learning
  - 02287 Logical Theories for Uncertainty and Learning

- **Natural language processing** (Ch. 21-22 of R&N)
  - 02281 Data Logic
  - 02456 Deep Learning (some)

Other AI-relevant courses at DTU

- **Image processing** (Ch. 24 of R&N)
  - 02502 Image Analysis
  - 02506 Advanced Image Analysis
  - 02504 Computer Vision
- **Robotics** (Ch. 25 of R&N)
  - 31380 Intelligent Systems
  - 31385 Autonomous Robot Systems
  - 31388 Advanced Autonomous Robots
  - 31389 Advanced Topics in Robotics and Autonomous Systems

It is a **master course**…

- Some exercises and assignments are more free and open-ended than in standard bachelor courses.
- Some exercise descriptions are less spelled out, requiring more from the reader.
- The course requires both high mathematical maturity and solid programming experience.
- You are expected to be able to independently search for relevant/additional literature.
- You are expected to independently be able to manage your time (in particular **extremely important** in the programming project).

Today's subject:
AI search basics

Essentially supposed to be repetition from
02180 Introduction to Artificial Intelligence.

GRAPH-SEARCH algorithm

Based on Figure 3.7 of Russell & Norvig

function GRAPH-SEARCH (*problem*) **returns** a solution, or failure
  exploredNodes := Ø
  initialize the frontier to be the initial state of the *problem*
  loop do
    if frontier is empty **then return** failure
    choose a leaf node n (that is, a node from the frontier)
    remove *n* from frontier
    if *n* is a goal state **then return** solution
    exploredNodes := exploredNodes ∪ { *n* }
    expand the node *n* (that is, compute its children)
    for each child *m* of *n*
      if *m* is not in frontier **and** *m* ∉ exploredNodes **then**
        add child m to frontier

GRAPH-SEARCH in Java

`SearchClient.java` in code of Assignment 1

```
strategy.addToFrontier( this.initialState );
[…]
while ( true ) {
[…]
if ( strategy.frontierIsEmpty() ) {
    return null;
}

Node leafNode = strategy.getAndRemoveLeaf();

if ( leafNode.isGoalState() ) {
 return leafNode.extractPlan();
}

strategy.addToExplored( leafNode );
for ( Node n : leafNode.getExpandedNodes() ) {
  if ( !strategy.isExplored( n ) && !strategy.inFrontier( n ) ) {
  strategy.addToFrontier( n );
  }
}
[…]
```

Different (uninformed) search strategies with GRAPH-SEARCH

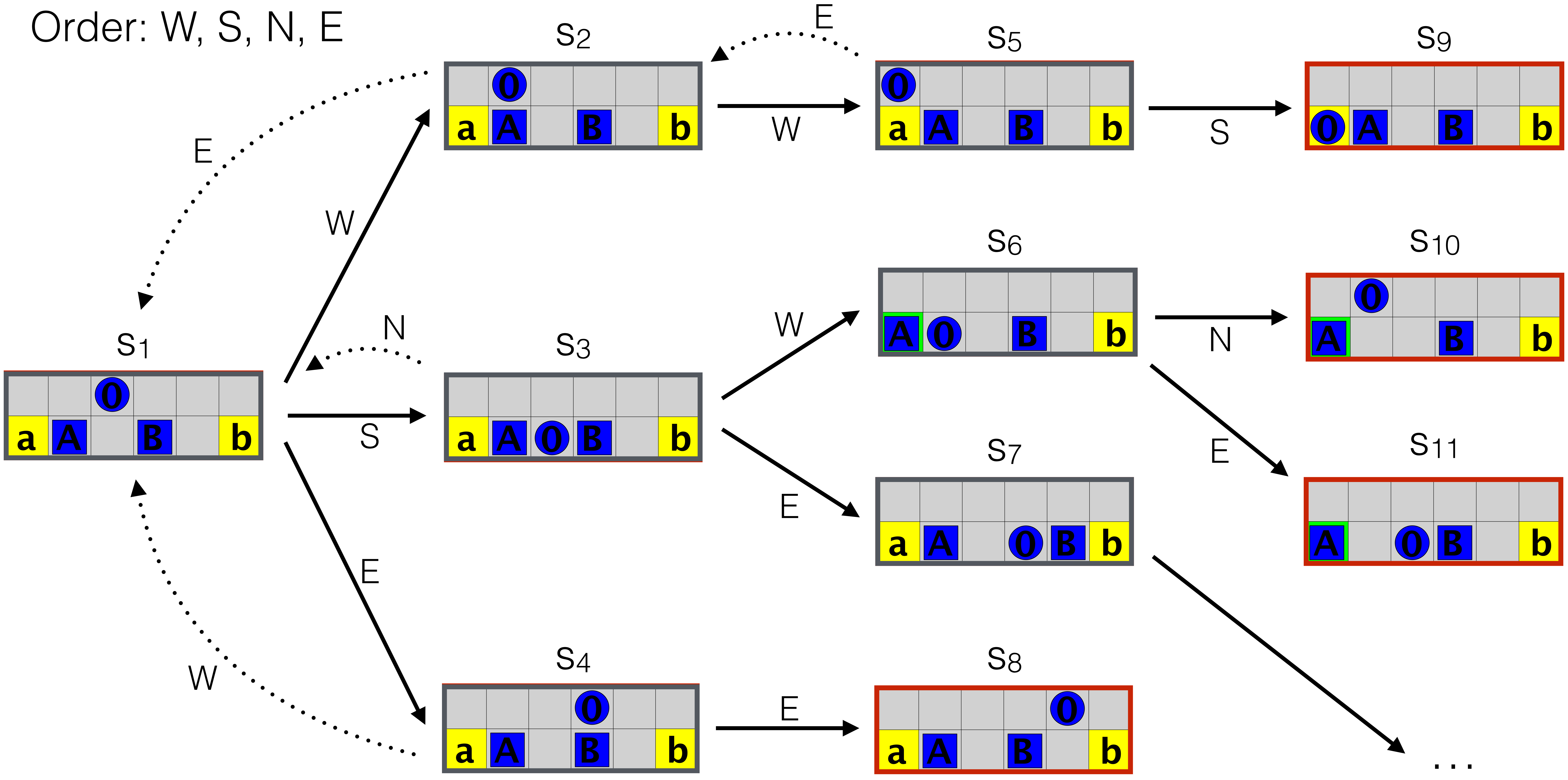
Different search strategies can be achieved by simply changing how **choose leaf node** and **add child to frontier** work.

- **Breadth-First Search (BFS):**
  - Frontier is a **queue** (FIFO).
  - **Choose leaf node**: dequeue node from frontier.
  - **Add child to frontier**: enqueue node to frontier.

- **Depth-First Search (DFS)**:
  - Frontier is a **stack** (LIFO).
  - **Choose leaf node**: pop node from frontier.
  - **Add child to frontier**: push node to frontier.

BFS example (Sokoban rules = only pushing)

Order: W, S, N, E



Red border: in frontier. **Grey border**: in exploredNodes.

DFS example (Sokoban rules = only pushing)

Order: W, S, N, E

Red border: in frontier. **Grey border**: in exploredNodes.

Informed search strategies with GRAPH-SEARCH

- **Best-First Search:**
  - Frontier is a **priority queue** where the key of a node n is denoted $f(n)$.
  - **Choose leaf node**: extract minimal node from priority queue (that is, a node n with minimal $f(n)$).
  - **Add child to frontier**: insert node into priority queue.

- Let $g(n)$ denote the distance of a node n from the initial state (length of shortest path from init state to n). Let $h(n)$ denote a **heuristic function** ($h(n)$ gives an estimate of the distance from n to the goal).

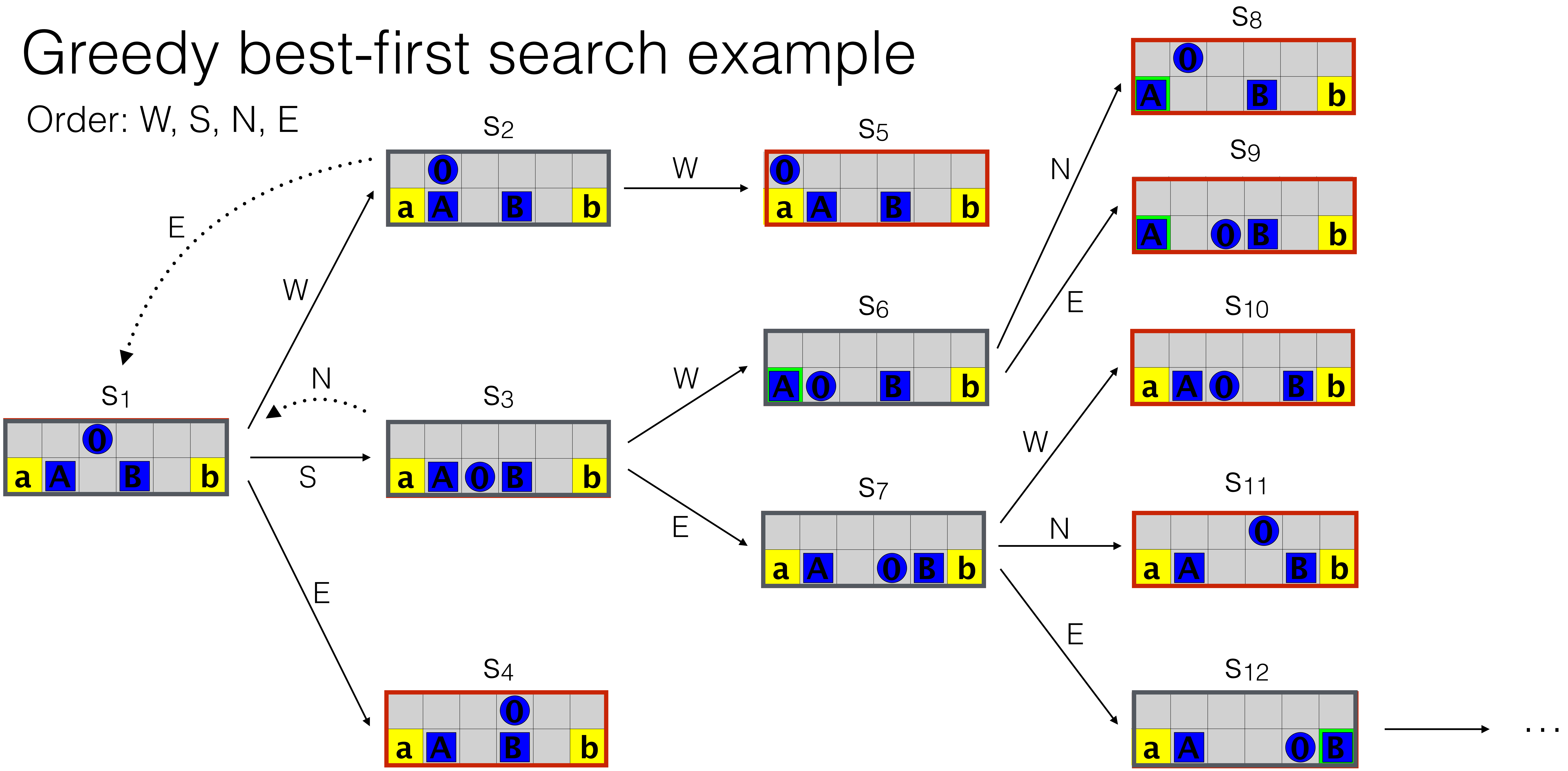
- Different versions of best-first search depending on how the key $f(n)$ of a node n is calculated:
  - **A***: $f(n) = g(n) + h(n)$.
  - **WA***: $f(n) = g(n) + W\, h(n)$, for some constant $W > 1$.
  - **Greedy best first search**: $f(n) = h(n)$.
  - Which algorithm do you get when letting $f(n) = g(n)$?

Designing heuristic functions

- Designing good heuristic functions *h(n)* is a great art form.
- Repetition from 02180 (R&N Ch. 3): What properties should a good heuristic function have?
- You will work with designing heuristic functions both in Assignment 1 and 3.
- A simple example for the Sokoban domain assuming unique goals for each box:
  h(n) = the sum of the distances of boxes to their goals.
- See example using this heuristics on the next page.

Greedy best-first search example

Order: W, S, N, E



Red border: in frontier. **Grey border**: in exploredNodes.

Exercise session today:
Start working on Assignment 1.

Deadline in a little less than two weeks.