Intelligent Systems
Summer Term 2025
2. Assignment
Description Logic, Constraint Programming, Planning.

Submission due **28.6.2025/23:55** via TeachCenter.
Solutions generated using AI tools like ChatGPT are not allowed!

Task 1 (Description Logic), 10 points

a) Semantics (2 points) For the following symbols:

$$\begin{array}{ccc} \square & \bigstar & \spadesuit \\ \clubsuit & \bigcirc & \diamondsuit \\ \heartsuit & \blacksquare & \triangle \end{array}$$

Consider the interpretation $\mathcal{I}$ defined as:

- $\Delta^{\mathcal{I}} = \{\square, \bigstar, \spadesuit, \clubsuit, \bigcirc, \diamondsuit, \heartsuit, \blacksquare, \triangle\}$,
- $\text{Suit}^{\mathcal{I}} = \{\spadesuit, \clubsuit, \diamondsuit, \heartsuit\}$,
- $\text{White}^{\mathcal{I}} = \{\square, \bigcirc, \diamondsuit, \heartsuit, \triangle\}$,
- $\text{Black}^{\mathcal{I}} = \{\bigstar, \spadesuit, \clubsuit, \blacksquare\}$,
- $\text{over}^{\mathcal{I}} = \{(\square, \clubsuit), (\bigstar, \bigcirc), (\spadesuit, \diamondsuit), (\clubsuit, \heartsuit), (\bigcirc, \blacksquare), (\diamondsuit, \triangle)\}$,
- $\text{left}^{\mathcal{I}} = \{(\square, \bigstar), (\bigstar, \spadesuit), (\clubsuit, \bigcirc), (\bigcirc, \diamondsuit), (\heartsuit, \blacksquare), (\blacksquare, \triangle)\}$,

Prove whether $\mathcal{I}$ satisfies:

1. $\text{White} \sqcap \text{Suit} \sqcap \exists\, \text{over}\, .\text{White} \sqsubseteq \neg \exists\, \text{over}\, .\text{Black}$
2. $\exists\, \text{over}\, .\text{Black} \equiv (\neg\, \text{Suit} \sqcap \forall\, \text{left}\, .\top) \sqcap \neg\, \text{Black}$

b) Reasoning (2 points) Given the following description logic domain:
TBOX:

- $Male \equiv \neg Female$
- $Mother \equiv Female \sqcap \exists isParentOf.Person$
- $Father \equiv Male \sqcap \exists isPArentOf.Person$
- $Parent \equiv Mother \sqcup Father$

ABOX:

- $Person(BOB)$
- $isParentOf(ANNE, BOB)$
- $\neg Male(ANNE)$

Answer the following query using the Tableaux Algorithm:

- $Mother(ANNE)$

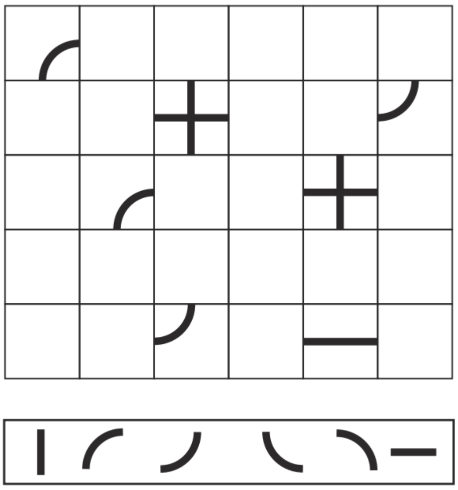

Figure 1: Track layout for Task 2. The available track segments are shown below.

c) Modeling and Reasoning (6 points) Model an ontology of meal preparation. A meal is defined by a recipe that specifies what ingredients, tools, and appliances are needed to prepare the meal. If applicable, please model if an ingredient or tool can be substituted by some other item. For instances of the domain, please model the availability of tools, appliances, and ingredients.

Implement your model using OWL 2 DL in Protégé [2]. Represent at least 3 recipes, 6 ingredients, and 5 tools in your model. Model 3 different instances of inventory and provide the related queries using the Manchester OWL Syntax [1] [2] which show that your model can reason about what meals can be prepared in what situation. If needed, you may extend your model with SWRL Rules [3].

Task 2 (Constraint Programming), 8 points

a) Modeling and Solving (4 points) In the railway track challenge, the problem is to build one continuous track (without discontinuities) by placing straight and curved sections while some sections are already fixed. A section can be used several times.
Model the railway track problem shown in Figure 1 as a constraint satisfaction problem using the *MiniZink* language. Solve the above problem instance using the *MiniZink* IDE [4].

b) Modeling and Solving (4 points) Given 4 agents and 6 tasks. Each task has an individual duration and consumes an amount of resources if executed. Based on its remaining resources, an agent is able to perform only particular

[1] https://protegewiki.stanford.edu/wiki/DLQueryTab
[2] https://www.w3.org/TR/owl2-manchester-syntax/
[3] https://www.w3.org/submissions/SWRL/
[4] https://www.minizinc.org/

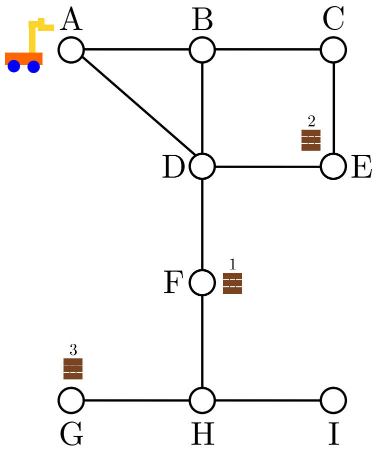

Figure 2: Delivery robot planning domain for Task 3. The robot can traverse the network by moving from one node to a neighboring node. The robot can transport several packages, but has a maximum capacity for carrying. It can load a package (packages are identified with numbers and have a particular weight) if it is in the same position as the robot and it does not extend the capacity. The robot can unload a package if it carries it and it is on the target location. It can transport loaded packages between different locations (locations are identified with letters). Transfers between nodes are instantaneous.

tasks. Resources of agents get consumed and will not be refilled. If an agent executes a task, it consumes resources equal to the task's duration.

Duration of Tasks:

Task	1	2	3	4	5	6
Duration	37	78	11	56	90	43

Agent Resources:

Agent	1	2	3	4
Available Resources	123	103	67	55

Model the assignment problem as a constraint satisfaction problem using the *MiniZink* language. Solve the above problem instance using the *MiniZink* IDE.

Task 3 (Planning), 12 points

a) PDDL encoding and solving (6 points) The Planning Domain Definition Language [4, 1, 3] (PDDL) is used for a large community of researchers to specify planning models. There are different versions and extensions that increase the expressivity of the language. Nevertheless, not all planners can handle the full specification. Considering the domain and constraints shown in Figure 2:

1. Model the planning domain and problem in PDDL 2.1.

2. Find a plan using the PDDL online editor [6] or Visual Studio Code with the PDDL extension [5]. For both option we recommend using the *optic* planner. The initial configuration is that shown in Figure 2. The goal is to move packages 1 and 2 to position H and package 3 to B, and the robot should return to location A when finished. The maximum capacity of the robot is 50, while the weights of packages 1, 2, and 3 are 20, 10, and 50. Try different initial configurations and goals to check the correctness of your model. **(2 points)**

b) alternative encoding (6 points) Model the above planning problem as a constraint satisfaction problem using the *MiniZink* language [5]. Use step-indexed variables for action execution and state information, and model action precondition and effects as constraints. The initial state and the goal are also expressed by constraints on the state variables. The frame problem needs to be addressed as well. Solve the above problem instance using the *MiniZink* IDE.

Submission requirements

The following files should be submitted via TeachCenter:

1. Report. A single PDF file containing the solution to this assignment. The report should explain the solution to all questions.

2. Models. In addition to the PDF report, the following model files should be uploaded:

   (a) Task 1.c. 3 ontology (OWL) file: *task1c_{1..3}.owl*.

   (b) Task 2.a. Minizinc file named *task2a.mzn*.

   (c) Task 2.b. Minizinc file named *task2b.mzn*.

   (d) Task 3.a. PDDL domain and problem files named *task3a-domain.pddl* and *task3a-problem.pddl* respectively.

   (e) Task 3.b. Minizinc file named *task3b.mzn*.

References

[1] Fares Alabound and Andrew Coles. LearnPDDL: A guide to learning, implementing and using PDDL. `https://fareskalaboud.github.io/LearnPDDL/`. Accessed: 2025-05-05.

[2] Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. Protégé. `https://protege.stanford.edu/`. Accessed: 2025-05-05.

[3] Maria Fox. Pddl2.1 : An extension to pddl for expressing temporal planning domains. `https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume20/fox03a-html/JAIRpddl.html`. Accessed: 2022-04-26.

[5] `https://code.visualstudio.com/`

[4] Adam Green, Benjamin Jacob Reji, Felipe Meneguzzi, Francisco Martin Rico, Henry Stairs, Jan Dolejsi, Mau Managuagno, and Mounty Jonhathan. Planning.Wiki – The AI Planning & PDDL Wiki . `https://planning.wiki/`. Accessed: 2025-05-05.

[5] Adriana Lopez and Fahiem Bacchus. Generalizing graphplan by formulating planning as a csp. In *IJCAI*, volume 3, pages 954–960, 2003.

[6] Christia Muise and Trello Board. PDDL Editor. `http://editor.planning.domains`. Accessed: 2022-04-26.