COL750: Foundations of Automatic Verification

Assignment

Due Date and time: 29/03/2025, 18:00

Please check the course webpage for late submission policies. There will be zero tolerance for dishonest practices, including copying or sharing solutions/code or use of OpenAI tools in the assignment. Any violation will result in an immediate F grade and will be reported to DisCo

Submission Instructions Your final submission should include the following:

Source Code
1. Your source code for solving the puzzle using a SAT solver. The program should be able to take input in a specified format and produce output as described.
2. Your source code for solving the puzzle using an SMT solver. The program should be able to take input in a specified format and produce output as described.
3. Both source codes must include a flag to incorporate incremental solving.

NuSMV File Your .smv file for execution on the NuSMV tool.

Report Maximum 4 Pages, Excluding References. The report must include:

1. The CNF encoding of the puzzle.
2. The SMT encoding of the puzzle.
3. Explanation of the .smv File: a description of the NuSMV model and its purpose.

README A detailed README file explaining how to run your codes.

Submission Format Package all the required files (source codes, .smv file, report, and README) into a ZIP folder. Name the folder as Entrynumber _Firstname_Lastname.zip. Submit the ZIP folder on **Gradescope**.

Finding a Safe Path – SAT and Z3 Encoding

Imagine that you are trapped in a hazardous $N \times N$ grid and must navigate from the upper left corner $(1, 1)$ to the bottom right corner (N, N). However, the grid is filled with obstacles and you can only move right ($\rightarrow$) or down ($\downarrow$). Your goal is to find a safe path while ensuring that each step adheres to the given constraints.

- Your program should be able to receive input in the format shown in Table 1, where **S** represents the starting position $(1, 1)$, and **G** denotes the goal position (N, N). The input specifies the grid structure, including obstacles **X**, which must be avoided while determining a valid path.

S	.	**X**	.	.
.	**X**	.	**X**	.
.	.	.	**X**	.
X	**X**	.	.	.
.	.	.	**X**	**G**

Table 1: 5×5 Grid with Obstacles

- The output of your program should be **SAT** along with a valid path from S to G if such a path exists. If no valid path can be found, the program should return **UNSAT**.
- You need to write a program that takes this input and converts the path constraints—such as starting from the given position, avoiding obstacles, and only allowing right ($\rightarrow$) and down ($\downarrow$) moves—into a **SAT instance**. Specifically, you should encode these constraints in CNF (Conjunctive Normal Form) and then use a SAT solver to determine if a valid path exists, and return the output.

Question 1 You need to write a program that takes this input and converts the path constraints—such as starting from the given position, avoiding obstacles, and only allowing right ($\rightarrow$) and down ($\downarrow$) moves—into a **SAT instance**. Specifically, you should encode these constraints in CNF (Conjunctive Normal Form) and then use a SAT solver to determine if a valid path exists, and return the output. You are allowed to use PySAT APIs whenever needed.

Question 2 You need to write a program that takes this input and converts the path constraints—such as starting from the given position, avoiding obstacles, and only allowing right ($\rightarrow$) and down ($\downarrow$) moves—into a **SMT instance**. You are allowed to use any underlying theory except Boolean and Bitvectors and then use a SMT solver to determine if a valid path exists, and return the output. You are allowed to use Z3 APIs whenever needed.

Question 3 Solve the problem **incrementally** using SAT solvers (question 1) by restricting the number of moves to be at most $2N$. Ensure that the solution adheres to the step limit while still allowing a valid path from S to G, if one exists.

The Traffic Intersection Problem

A busy four-way intersection has two traffic lights that control the flow of cars from two perpendicular roads (North-South and East-West). Each road has two lanes (one for each direction), and cars follow simple traffic rules. The traffic light system ensures that:

- Only one road (either North-South or East-West) can have a green light at a time.
- The green light stays on for a fixed duration, after which it turns red, and the other road gets a green light.
- There is a brief transition state (yellow light) before switching between green and red.

Each car follows a cyclic behavior:

- Waiting: If the light is red, the car must stop at the intersection.
- Passing: If the light is green and there is no other car in front, the car moves forward.
- Cleared: After passing the intersection, the car leaves the system.

Your task is to model this system in NuSMV and verify the following properties:

- Liveness: Every car will eventually get a green light and pass through the intersection (no car is stuck forever).
- Fairness: The traffic light system must alternate between North-South and East-West roads infinitely often.

Question 4 Define the system in NuSMV using state transitions. Specify properties in CTL (Computation Tree Logic) to ensure the system satisfies fairness and liveness. Verify the model using NuSMV and check if it satisfies the given requirements.