Automated Reasoning IMC009
Practical Assignment – Part 2

The assignment

Each part of the practical assignment has to be executed by one or two persons. (It is not mandatory to use the same group as for part 1.) The result of each part should be described in a report that should be submitted in PDF via Brightspace, preferably less than 20 pages.

For all used formulas an extensive documentation is required, explaining the approach and the overall structure. A generic approach is preferred, since this may result in clearer descriptions, increasing the confidence in the correctness of the results. Formulas of more than half a page should not be contained in the report; instead the structure of the formula should be explained. From the output of the programs relevant parts should be contained in the report, and observations on computation time should be reported. The answers on the problems should be motivated, where relevant including pictures are appreciated. Every report should contain name, student number and email address of each of the authors. In case of two authors each of them is considered to be responsible for the full text and all results.

Grading

- Clear and generic descriptions are appreciated, both of the formulas themselves and the way they were designed.
- To obtain a 7, at least 3 out of the four solutions should be correct.
- Not giving a solution at all for one problem is preferred over giving a wrong solution.
- Reasons for obtaining higher than a 7 may be:
  - all problems correctly solved,
  - remarkably clear and structured descriptions,
  - original approaches and solutions,
  - a good answer on the bonus question (especially if it's better than others in the class),
  - a particularly interesting problem for question 4.

The programs to be used

- `Z3`: `https://github.com/Z3Prover/z3` .
- `Yices`: `http://yices.csl.sri.com/`.
- `cvc5`: `https://cvc5.github.io/`.
- `Prover9` and `Mace4`: `https://www.cs.unm.edu/~mccune/prover9/download/`

Each of the problems should be solved using one of these tools. The tools should do the job, but it may be necessary to use multiple calls, or add extra information to an implementation. No answer requires running a tool for more than 1 minute.

1 Groups

In mathematics, a *group* is defined to be a set G with an element $I \in G$, a binary operator $*$ and a unary operator *inv* satisfying

$$x * (y * z) = (x * y) * z, \quad x * I = x \quad \text{and} \quad x * inv(x) = I,$$

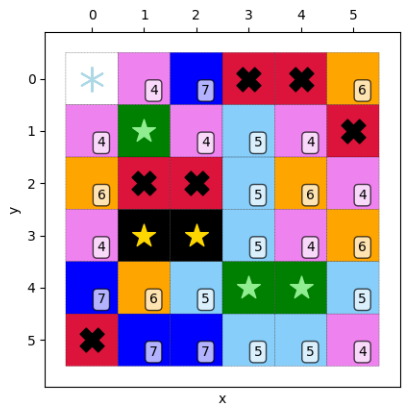
for all $x, y, z \in G$.
(a) Determine whether in every group each of the four properties

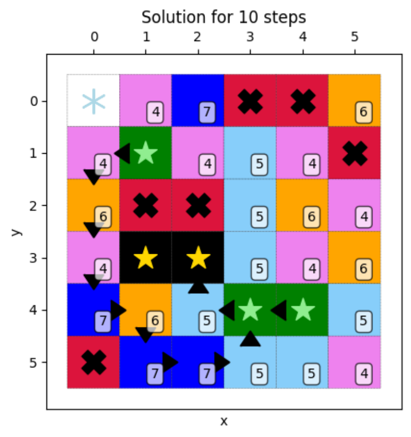
$$I * x = x, \quad inv(inv(x)) = x, \quad inv(x) * x = I \quad \text{and} \quad inv(x * y) = inv(x) * inv(y)$$

holds for all $x, y \in G$. If a property does not hold, determine the size of the smallest finite group for which it does not hold.
(b) A group is called *Abelian* if $x * y = y * x$ for all $x, y \in G$. Find the size of the smallest finite group that is not Abelian.
(c) Define $x^1 = x$ and $x^{n+1} = x * x^n$ for $n \geq 1$. For $n = 2, 3, 4$ do the following. Establish whether every group G satisfying $x^n = I$ for all $x \in G$ is Abelian. If not, determine the size of the smallest corresponding non-Abelian finite group.

2 Robot Planning

Consider the NxN grids below.





A robot starts in one of the green cells with a star. The goal is to lead the robot to one of the black cells marked with a star in as few steps as possible. That is, we want to limit the length of the trajectory from start to target in the worst-case (over all starting positions) by a number X. It suffices to solve the decision problem:

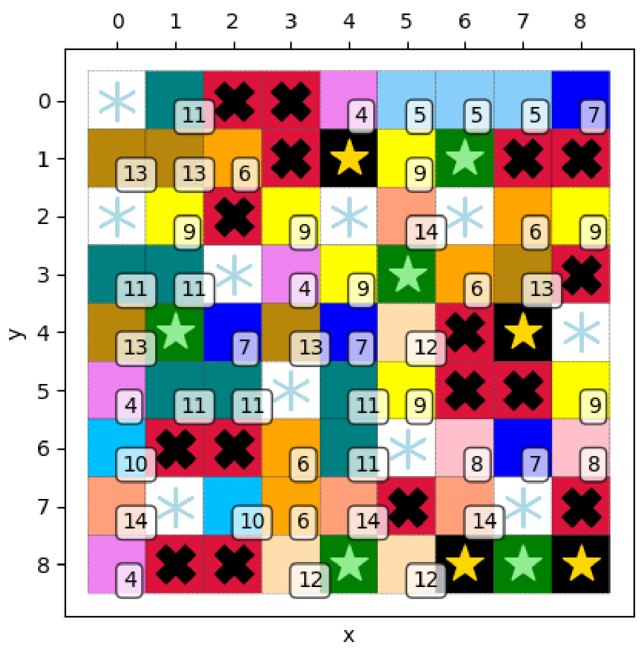
> Given a value for X, find an encoding so that we will reach a target within X steps from every `start` cell.

The witness for the answer is a plan that maps every cell to one of the possible directions. More precisely, the rules are as follows:

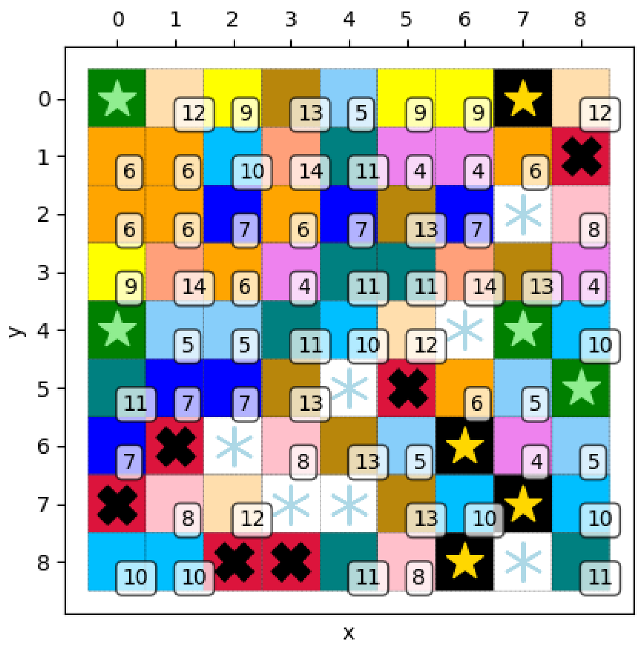
- A robot can move in one of the four cardinal directions (up, down, left, right). It will then reach the adjacent cell. It is allowed to stand still on an end point.
- Robots cannot fall of the grid. Instead, if they move towards a boundary, they will just stay where they are.
- The robot does not know where it is, but it can detect the surface type. Therefore, in every cell with the same surface type (described by a number/color), the robot must move in the same direction.
- A robot will catch fire if it visits a red `lava` cell (with a black cross). This makes it impossible to reach the end.
- On `ice` patches (white with a blue star), the robot that starts to move into a direction may accidentally move two cells instead of one. This is beyond your control, and you must reach the endpoint in X steps no matter what happens.

Please describe your encoding and describe the solutions for the following grids. The grids are also online as CSV files, where we use a zero for starts, a one for targets, a two for lava and a three for ice.

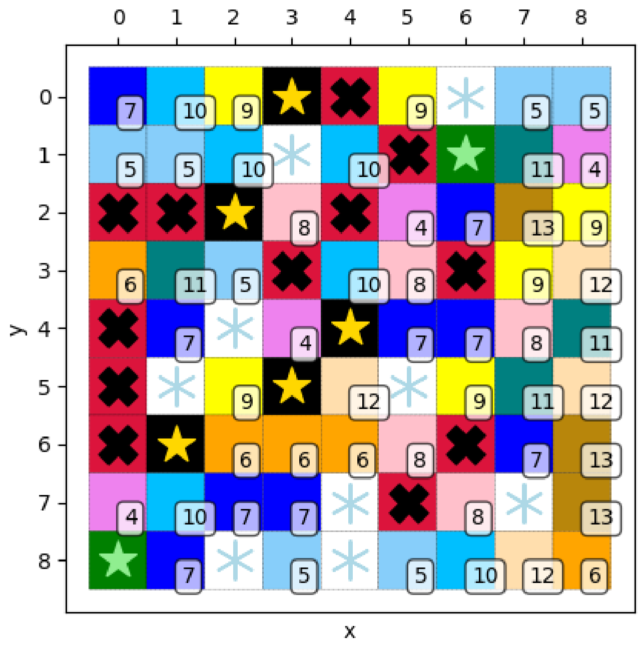
Grids


0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
x
y
11 4 5 5 5 7
13 13 6 9
9 9 14 6 9
11 11 4 9 6 13
13 7 13 7 12
4 11 11 11 9 9
10 6 11 8 7 8
14 10 6 14 14
4 12 12

(a)


0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
x
y
12 9 13 5 9 9 12
6 6 10 14 11 4 4 6
6 6 7 6 7 13 7 8
9 14 6 4 11 11 14 13 4
5 5 11 10 12 10
11 7 7 13 6 5
7 8 13 5 4 5
8 12 13 10 10
10 10 11 8 11

(b)


0
1
2
3
4
5
6
7
8
x
y
7
10
9
9
5
5
5
5
10
10
11
4
8
4
7
13
9
6
11
5
10
8
9
12
7
4
7
7
8
11
9
12
9
11
12
6
6
6
8
7
13
4
10
7
7
8
13
7
5
5
10
12
6

(c)

3 The Multiset Path Ordering

There are many extensions and variations of the recursive path ordering we have discussed in the lecture.[1] We will here discuss the *Multiset Path Ordering* (MPO). This is an ordering on terms that is generated from a *total quasi-ordering* relation $\trianglerighteq$ on function symbols, using multiset comparison.

Specifically, we are given a relation $\trianglerighteq$ on function symbols such that for all symbols $\mathtt{f}, \mathtt{g}$, we have: $\mathtt{f} \trianglerighteq \mathtt{g}$ or $\mathtt{g} \trianglerighteq \mathtt{f}$. Here, $\trianglerighteq$ is transitive (if $\mathtt{f} \trianglerighteq \mathtt{g} \trianglerighteq \mathtt{h}$ then $\mathtt{f} \trianglerighteq \mathtt{h}$) and reflexive (always $\mathtt{f} \trianglerighteq \mathtt{f}$). We denote $\mathtt{f} \triangleright \mathtt{g}$ if $\mathtt{f} \trianglerighteq \mathtt{g}$ and not $\mathtt{g} \trianglerighteq \mathtt{f}$. We denote $\mathtt{f} \equiv \mathtt{g}$ if $\mathtt{f} \trianglerighteq \mathtt{g}$ and $\mathtt{g} \trianglerighteq \mathtt{f}$.

The relations $\succsim$, $\succ$ and $\approx$ on terms are given by the following inductive definition:

1. $s \succsim t$ if and only if $s \succ t$ or $s \approx t$
2. $s \succ t$ if and only if $s = \mathtt{f}(s_1, \dots, s_n)$ and at least one of the following holds:
   (a) $s_i \succeq t$ for some $i \in \{1, \dots, n\}$
   (b) $t = \mathtt{g}(t_1, \dots, t_m)$ with $\mathtt{f} \triangleright \mathtt{g}$ and $s \succ t_i$ for all $i \in \{1, \dots, m\}$
   (c) $t = \mathtt{g}(t_1, \dots, t_m)$ with $\mathtt{f} \equiv \mathtt{g}$ and $\{\{s_1, \dots, s_n\}\} \succ_{\text{mul}} \{\{t_1, \dots, t_m\}\}$
3. $s \approx t$ if and only if one of the following holds:
   (a) s and t are both the same variable;
   (b) $s = \mathtt{f}(s_1, \dots, s_n)$ and $t = \mathtt{g}(t_1, \dots, t_n)$ and $\mathtt{f} \equiv \mathtt{g}$ and $\{\{s_1, \dots, s_n\}\} \approx_{\text{match}} \{\{t_1, \dots, t_n\}\}$

Here, a *multiset* is a set that may contain repetitions; order does not matter, but count does. For example, $\{\{1, 1, 2, 3\}\} = \{\{1, 3, 2, 1\}\}$ but this is not the same as $\{\{1, 2, 3, 3\}\}$. For two multisets A, B we say that $A \approx_{\text{match}} B$ if and only if we can write $A = \{\{s_1, \dots, s_n\}\}$ and $B = \{\{t_1, \dots, t_n\}\}$ and each $s_i \approx t_i$ (note that we can order the elements of A and B in any way!). We say that $A \succ_{\text{mul}} B$ if we can write $A = A_1 \cup A_2$, $B = B_1 \cup B_2$, A_2 is non-empty, $A_1 \approx_{\text{match}} B_1$ and for all $b \in B_2$ there is $a \in A_2$ such that $a \succ b$.

(a) Find a symbol ordering such that the following inequalities all hold, where x, y, z, u, v are all variables, and all other symbols are functions.[2]

- $\mathtt{c}(x, y, u, v) \succ \mathtt{b}(\mathtt{f}(x, y), \mathtt{b}(u, u, u), \mathtt{g}(v, \mathtt{b}(x, y, u)))$
- $\mathtt{b}(\mathtt{f}(x, y), \mathtt{g}(x, y), \mathtt{f}(x, \mathtt{g}(z, u))) \succ \mathtt{b}(\mathtt{f}(x, z), y, \mathtt{g}(\mathtt{g}(\mathtt{g}(y, x), x), x))$
- $\mathtt{h}(\mathtt{g}(x, \mathtt{g}(u, z)), \mathtt{c}(x, y, x, z)) \succ \mathtt{f}(\mathtt{d}(x, z), u)$
- $\mathtt{h}(\mathtt{d}(\mathtt{f}(x, y), \mathtt{g}(u, v)), \mathtt{f}(x, y)) \succ \mathtt{f}(\mathtt{c}(u, x, v, y), \mathtt{g}(y, x))$
- $\mathtt{f}(\mathtt{b}(x, y, z), u) \succ \mathtt{h}(u, \mathtt{f}(x, \mathtt{h}(y, x)))$
- $\mathtt{b}(\mathtt{a}(x, y, z), y, x) \succ \mathtt{c}(x, x, y, x)$

[1]We advise not doing this exercise until after RPO was discussed in the lectures! This will be done in the third lecture of the second quarter.

[2]If you want to write a parser for terms, note that (1) all function symbols take at least two arguments, (2) both variables and function symbols are a single letter, and (3) there is always a space after a command, and never after a bracket unless the term has ended.

Provide the symbol ordering, and give the (human-readable) proof for the third inequality.

(b) Can the inequalities be oriented with any other symbol ordering?

Tip: for ordered sequences $s_1, \dots, s_n$ and $t_1, \dots, t_m$, we have $\{\{s_1, \dots, s_n\}\} \succ_{\text{mul}} \{\{t_1, \dots, t_m\}\}$ if there exist functions $\mathsf{EQ} : \{1, \dots, n\} \to \{\top, \bot\}$ and $\varphi : \{1, \dots, m\} \to \{1, \dots, n\}$ such that:

- for all $i \in \{1, \dots, m\}$:
  - if $\mathsf{EQ}(\varphi(i)) = \bot$ then $s_{\varphi(i)} \succ t_i$;
  - if $\mathsf{EQ}(\varphi(i)) = \top$ then $s_{\varphi(i)} \approx t_i$;
  - if $\mathsf{EQ}(\varphi(i)) = \top$ then i is the only index mapping to $\varphi(i)$; that is, there is no $i' \in \{1, \dots, m\} \setminus \{i\}$ with $\varphi(i) = \varphi(i')$;
- there is at least one $j \in \{1, \dots, n\}$ with $\mathsf{EQ}(j) = \bot$

That is, $\mathsf{EQ}(j)$ indicates that $s_j \in A_1$. We have $\{\{s_1, \dots, s_n\}\} \approx_{\text{match}} \{\{t_1, \dots, t_m\}\}$ if $n = m$ and there exists a function $\varphi : \{1, \dots, m\} \to \{1, \dots, n\}$ such that for all $i \in \{1, \dots, m\}$:

- $s_{\varphi(i)} \approx t_i$
- for all $i' \in \{1, \dots, m\} \setminus \{i\}$: $\varphi(i') \neq \varphi(i)$.

4 Your own problem!

Give a precise description of a non-trivial problem of your own choice, and encode this and solve it by one of the given programs.

Here ‘trivial problems’ include both minor modifications of earlier problems and single solutions of simply specified puzzles like sudokus. In case of doubt please contact the teaching assistant.