Project Description

GitHub: https://github.com/uni-aau/asp-project
Clingo: https://potassco.org/clingo/run/
Puzzle Visualizer: https://puzzle-visualizer.duckdns.org/

Implementation

I first define the facts of my ASP program. The facts correspond to the `cell/3` predicate, which consists of the following structure: `cell(ROW, COLUMN, SYMBOL).` For example, a fact might look like `cell(5,5,"a").` I took the examples from ASP chef.

I then first defined the dimensions of the grid.
To do that, I calculated the max amounts of rows and columns and then define the range of rows and columns in the grid.

```
% Dimensions of grid
row(1..R) :- R = #max { Row : cell(Row, _, _) }.
col(1..C) :- C = #max { Col : cell(_, Col, _) }.
```

However, Clingo did not understand the #max syntax, therefore I used a more straightforward approach where I just define the predicate row/1 and col/1 by using the existing cell/3 predicate with the respective Row/Col variable.

```
row(Row) :- cell(Row, _, _).
col(Col) :- cell(_, Col, _).
```

Then I define that the solver can add a cell(Row, Col, Symbol) to my crossedout(Row, Col, Symbol) set. The solver will add specific cells to crossedout/3 that match my defined constraints below.

```
{ crossedout(Row, Col, Symbol) } :- cell(Row, Col, Symbol).
```

Constraint 1: No symbols appear more than once in any rows
Here I check, that if two cells are in the same row, with the same symbol and neither cell is crossed out, it violates the constraint and one of the two cells has to be crossed out.

```
:- cell(Row, Col1, Symbol), cell(Row, Col2, Symbol),
   Col1 != Col2,
   not crossedout(Row, Col1, Symbol), not crossedout(Row, Col2, Symbol).
```

I do the same for the columns. If two cells are in the same column with the same symbol, the constraint is violated if neither cell is crossed out.

```
:- cell(Row1, Col, Symbol), cell(Row2, Col, Symbol),
   Row1 != Row2,
   not crossedout(Row1, Col, Symbol), not crossedout(Row2, Col, Symbol).
```

Constraint 2: Crossed-out symbols must not occur next to each other horizontally or vertically

I first check if the manhatten distance of two cells is 1 to be able to determine in the next step which crossedouts are next to each other.

```
nextTo(Row1, Col1, Row2, Col2) :- row(Row1), row(Row2), col(Col1), col(Col2),
                                  |Row1 - Row2| + |Col1 - Col2| = 1.
```

In the second step I specify that if two crossed out cells are next to each other (no matter which symbol is used), the constrain is violated.

```
:- crossedout(Row1, Col1, _), crossedout(Row2, Col2, _), nextTo(Row1, Col1, Row2, Col2).
```

Constraint 3: All symbols that are not crossed out should form a contiguous area (considering horizontal and vertical connections).
So to simplify that, it means, that each non-crossed out symbol must be reachable from all other non-crossed out symbols by moving only horizontally or vertically over non-crossed out symbols

Firstly, I define reachable/4 where each not crossed row and column is added, since it is reachable from itself.

```
reachable(Row, Col, Row, Col) :- cell(Row, Col, _), not crossedout(Row, Col, _).
```

Then I mark all cells as reachable, if the cell next to another non-crossed out cell is also reachable. The symbol in constraint 3 is not relevant, that's why i used _ instead of actual symbols.

```
reachable(Row1, Col1, Row2, Col2) :- reachable(Row1, Col1, Row3, Col3), nextTo(Row3, Col3, Row2, Col2),
                                     not crossedout(Row2, Col2, _).
```

And after that, i ensure, that all non-crossed out symbols form a single connected component. If a non-crossed out cell is not reachable from any other non-crossed out cell, the constraint is violated

```
:- cell(Row, Col, _), not crossedout(Row, Col, _),
   not reachable(Row, Col, Row0, Col0), cell(Row0, Col0, _), not crossedout(Row0, Col0, _).
```

At the end I output the solution by showing crossedout/3

```
#show crossedout/3.
```

After that, I tested my implementation in clingo:

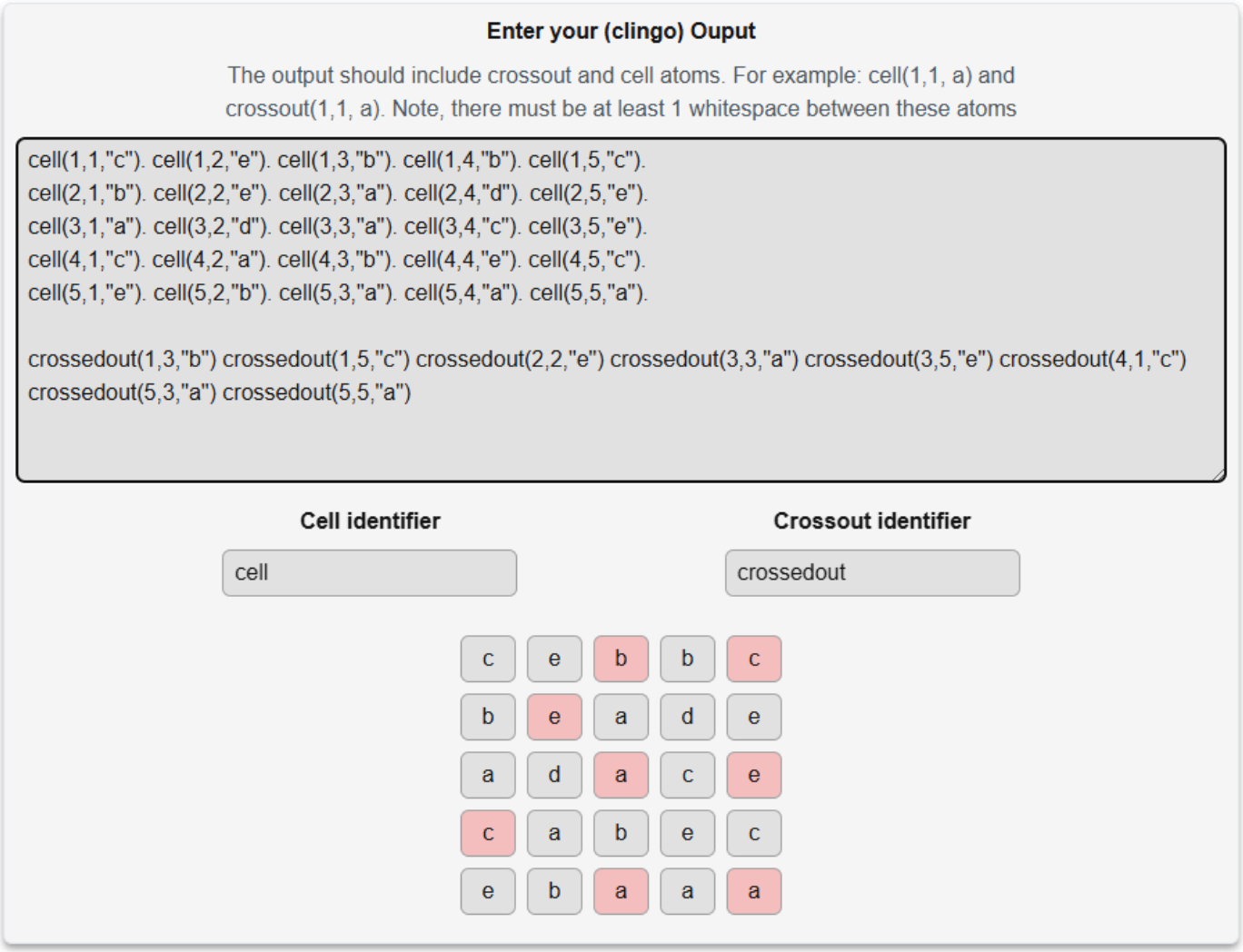
```
clingo version 5.7.0
Reading from stdin
Solving...
Answer: 1
crossedout(1,3,"b") crossedout(1,5,"c") crossedout(2,2,"e") crossedout(3,3,"a") crossedout(3,5,"e") crossedout
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 0.034s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

I receive:

```
crossedout(1,3,"b") crossedout(1,5,"c") crossedout(2,2,"e")
crossedout(3,3,"a") crossedout(3,5,"e") crossedout(4,1,"c")
crossedout(5,3,"a") crossedout(5,5,"a")
```

Checking via puzzle visualizer, you can see that all three defined constraints are successfully satisfied. Every non-crossed out cell is reachable, no double entries in row and column and also no double entries next to each other



Ethical Considerations:

The essay should also contain the following additional consideration on ethics: Imagine that a variant of your solver will be deployed not in a computer game, but in reality. Think whether there are any military settings in which such a tool could be useful. Starting from such a military scenario, discuss the deployment of AI methods in military applications from an ethical perspective. Providing this discussion on ethics is required for passing.

The cross-out puzzle in general seems quite trivial. However, there must not always be such trivial cases. In military, such problems can be used as a strategy to defend or attack against enemies. An example would be to find a path through a mine field where you only know some general information, such as for example a mine will not be next to another mine in a radius of 5 meters. You can then define the field as grid and create constraints as we did in our puzzle solver to find a path without triggering a mine.

Another example would be a tactical resource deployment optimization for a military base. While allocating resources to the bases, there are constraints like no multiple high-priority assets such as medical teams should be placed in the same region or each unit has access to at least one supply route. The network of supply chains should also remain connected to ensure no stranded units similar to the crossed-out puzzle where all non-crossed out fields should be accessible by the others. To find the best allocation of resources based on those example constraints, the use of ASP or similar applications can be used.

When AI-driven methods are deployed in military operations, a significant ethical concern arises due to the absence of direct human responsibility in critical decision-making. Additionally, AI still lacks of moral reasoning, empathy, and the ability to weigh ethical trade-offs in life and death decisions which increases the risk of actions that violate humanitarian principles and escalate conflicts with the enemy. Moreover, AI systems can also make mistakes due to misclassification and data bias, further amplifying the risk for bad and harmful decisions.