Automated reasoning project report

Roberto Tonino

October ??, 2024

1 Problem statement

Delle astronavi aliene sono disposte in modo complanare in modo che possiamo immaginarle nel rettangolo cartesiano (0,0), (max_x,0), (max_x,max_y), (0,max_y). max_x e max_y sono dati di input e vengono anche date le coordinate (intere) delle astronavi presenti. Supponiamo non siano già ad ordinata 0 (ovvero sulla terra).

Ci accorgiamo che ad ogni istante di tempo ogni astronave scende unitariamente (ciascuna componente y cala di 1).

Gli alieni sono ostili e vogliamo fermarli. Disponiamo di un cannone all'inizio in posizione (0,0) che in un istante di tempo può (1) sparare verticalmente distruggendo la astronave più vicina che si trovi nella stessa ascissa, oppure (2) spostarsi (senza sparare) in un'altra ascissa. Il tempo per spostarsi di 1,2,3,4, etc caselle sull'asse x è sempre unitario.

Si vuole trovare un piano (se esiste) per evitare che anche solo una astronave tocchi terra.

P.S. É possibile ci sia una soluzione furba, ad-hoc, per il problema. Non cerchiamo quella. Vogliamo codificarlo e fare risolvere al CP solver o al ASP solver.

2 Clingo model

In the first few lines we find the domain predicates. There are a couple of things to note. The time/1 predicate started from 0. This is to indiccate the initial state of the world and means that the model will have t+1 time steps. The x and y axis also start from 0. This is because the grid is a cartesian rectangle. The cannon predicate is arguably the most odd: its' written such that the model can have more than one cannon, but practically the model never takes into account this eventuality. The cannon predicate comes in this shape in order to be able to use a variable C in further predicates.

The alien/1 predicate comes with an identifier, which is simply an integer counter strarting from 1.

The `1 { ... } 1` line is a core part of the model. it states that for each time step except for the last one the cannon either moes to an X or shots an alien. The last time step is left our because of how the shoot predicate/action is defined further on: when the cannon shoots, an aliens is considered dead in the next time step.

The "initial state" predicate uses the at/4 predicate by positioning the cannon at the coordinates 0,0 as required in the problem statement.

The at/4 predicate is also a core part of the model. At any fiven time step, it represents the position on the grid of the cannon or of an alien.

The model contains the move, shoot actions. The move/3 action represents only the movement of the cannon. The requirement of each alien decreasing its y coordinate at every time step is met via an inertia rule.

The shoot/3 action involves both the cannon and an alien. The cannon shoots an alien A if at time T their x coordinates are equal.

Additionally, we are enforcing the fact that if the cannon shoots, it will stay in the same position in the next time step

The inertia rules involve aliens. If, at a ceratin point, an alien is dead it will remain dead and it will preserve its position on the grid. If an alien is not dead, its y coordinate will decrease at each time step.

The constraints avoid the model to accept a solution where the cannon has a y coordinate different than zero—at each time step. They also avoid a solution where an alien reaches y=0,

which would violate the requirements in the problem statement.

3 Minizinc model

In the Minizinc model, we define upfront the variables we need the model to assign values to. The model represents time steps as array indices. Like in the clingo model, the move_x and the shoot variables span in t time steps, compared to t+1 timesteps of the other decision variables.

The plannin concepts of precondition and effect are represented by a conjunction.

The concept of action is represented by a quantified boolean formula (QBF): <formula> with A being the set of aliens and T being 0..t timesteps. The actions are encoded as constraint in the model.

In the model we also find a constraint that enforces, foreach time setep, the move_x or the shoot actions to not be in their null state.

Inertia rules are represented by a QBF of the form: <formula>. The idea is to have a if-else-like structure of an imperative programming language.

Finally, the only constraint in the model prohibits the cannon to shoot to teh same alien twice. This can probably be removed by adding a precondition to the shoot action.