Checking Policies for Reactive Agents

Zeynep G. Saribatur
Institute of Logic and Computation, TU Wien
zeynep@kr.tuwien.ac.at

1 Problem Description

Consider an agent in a grid-cell environment of size $n \times n$ with obstacles. The agent is initially located in the upper left corner, and it can horizontally/vertically observe all the cells (until an obstacle), and move arbitrary number of steps in these directions. The goal of the agent is to find the missing person. However, the complete instance of the environment (i.e., where the person and the obstacles are located) is unknown. Thus, the initial state consists of several possible instances of the environment, in terms of the obstacles' and the person's location. Due to this uncertainty, computing a plan that can reach the goal in all possible instances would be troublesome.

We consider an alternative approach, where the agent is also provided with a *policy*, that picks the actions to execute depending on the current state. This policy is designed by the user to help with the decision-making of the agent, while it traverses the environment. Given such a policy, we want to know whether by following this policy, the agent would achieve the goal in all possible instances of the environment. In order to do this check, we generate all possible initial states and then search for a *witness* of failure, which is a plan, executed in some instance, that shows that following the given policy does not make it to the goal. This failure can be due to the agent getting stuck at a location or starting to loop (i.e., reaches a state more than once).

Example 1 *Fig. 1(a-b-c) shows some of the possible instances of the environment with the agent's initial location* $(1,1)$*, i.e.,* $rAt(1,1,0)$*. In these initial states, the agent observes the obstacle at* $(1,3)$ *and the reachable cells as shown in Fig. 1(d). Consider a policy (A) described as "move to the farthest point you observe (until you observe the person, then move to the person)". We want to check if a witness can be found within 3 time steps.*

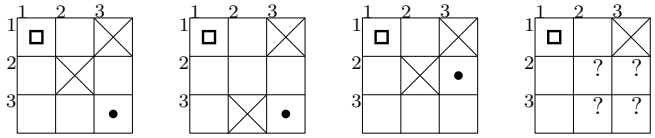

Figure 1: □:agent, •:person, ×:obstacle, ?:unknown

In (a), the person is found when the agent moves to the farthest observed point $(3,1)$*, i.e., no witness exists. In (b) and (c), we obtain a witness plan*

" $goTo(3,1,0), rAt(3,1,1), goTo(1,1,1), rAt(1,1,2), goTo(3,1,2), rAt(3,1,3)$ " *in which the agent comes back to* $(1,1)$ *at time step 2 and decides the same action to take as before (i.e., loops). Note that in (c), due to having two farthest points from* $(3,1)$ ($(1,1)$ *and* $(3,3)$)*, there is also the possible sequence of actions* " $goTo(3,1,0), goTo(3,3,1), goTo(2,3,2)$ "*, where the person is found. However, the witness shows the possibility to loop.*

The aim is to try different (complex) policies and see if/why they do not work. We expect the challenge of this problem to be large instances which makes the search and also the grounding difficult.

Predicates Below predicates are used to describe the problem:

$row(X), column(Y)$	rows and columns
$time(T)$	time domain
$pAT(X,Y)$	person is at cell X,Y
$obsAt(X,Y)$	cell X,Y has an obstacle
$rAt(X,Y,T)$	robot is at cell X,Y at time T
$goTo(X,Y,T)$	robot moving to cell X,Y at time T
$observed(X,Y,T)$	robot observes cell X,Y at time T
$farthest(X,Y,X_1,Y_1)$	cell X_1,Y_1 is the farthest from cell X,Y
$targetCell(X,Y,T)$	cell X,Y is picked to move to at time T
$reachable(X,Y)$	cell X,Y is reachable from (1,1)
$caught(T)$	robot reached the person at time T
$seen(T)$	person is seen at time T
$maxT(T)$	the maximum time step to reach the goal
$looped$	robot is looping or getting stuck at a position

The atom *looped* is always part of the policy encoding and is derived as true if and only if a failure occurred.

2 Input Description

The input consists of two rule sets (i1)-(i2) and the maximum time (i3) as described below. In the input file, they are marked and separated with comments in the format "%%%%% Input (ik) %%%%%", $k \in \{1,2,3\}$.

(i1) a set of possible instances of the environment (i.e., initial states), generated by a set of rules[1] where each answer set corresponds to a possible

[1]See Appendix and Section 4 for example definitions of *reachable*, *observed*, $farthest$, *seen* and *caught* used in the encodings.

instance:

$$row(1..n).column(1..n).$$
$$rAt(1,1,0).$$
% guess person's and obstacles' locations
$$\{pAt(X,Y) : row(X), column(Y)\} = 1.$$
$$\{obsAt(X,Y) : row(X), column(Y)\} = oNum.$$
% person can not be located at a not reachable cell
$$\bot \leftarrow pAt(X,Y), not\ reachable(X,Y).$$

n: environment size, $oNum$: total number of obstacles.

(i2) a description of a policy,
e.g., the policy described in Ex. 1 can be formulated as below[2].

$$\{targetCell(X_1,Y_1,T) : farthest(X,Y,X_1,Y_1)\} = 1 \leftarrow rAt(X,Y,T).$$
$$goTo(X,Y,T) \leftarrow targetCell(X_1,Y_1,T), not\ seen(T), not\ caught(T).$$
$$goTo(X,Y,T) \leftarrow seen(T), not\ caught(T), pAt(X,Y).$$
$$looped \leftarrow rAt(X,Y,T), rAt(X,Y,T_1), T_1 > T.$$

(i3) the maximum time step $maxT(t)$ and time domain $time$.

3 Output Description

At least one plan of failure consisting of $goTo$ and rAt, in an instance described by $row, column, pAt$, and $obsAt$. If none exists, returns UNSAT.

4 Example of One Solution

Consider the domain description as below.

$$timea(T-1) \leftarrow maxT(T).$$
$$timea(T-1) \leftarrow timea(T), 0 < T.$$
$$rAt(X,Y,T+1) \leftarrow goTo(X,Y,T), timea(T).$$
$$\neg rAt(X_1,Y_1,T) \leftarrow rAt(X,Y,T), row(X_1), column(Y_1), X_1 \neq X.$$
$$\neg rAt(X_1,Y_1,T) \leftarrow rAt(X,Y,T), row(X_1), column(Y_1), Y_1 \neq Y.$$
$$rAt(X,Y,T_1) \leftarrow not\ \neg rAt(X,Y,T_1), rAt(X,Y,T), T_1 = T+1, time(T_1).$$
$$\bot \leftarrow obsAt(X,Y), rAt(X,Y,T).$$
$$caught(T) \leftarrow rAt(X,Y,T), pAt(X,Y).$$
$$seen(T) \leftarrow observed(X,Y,T), pAt(X,Y).$$
$$caught(T+1) \leftarrow caught(T), timea(T).$$
$$seen(T+1) \leftarrow seen(T), timea(T).$$
$$\bot \leftarrow seen(T), not\ caught(T+1), timea(T).$$

Since we are searching for a plan that does not achieve the goal, where the agent loops, we give the following constraints.

$$\bot \leftarrow caught(T), maxT(T).$$
$$\bot \leftarrow seen(T), maxT(T).$$
$$\bot \leftarrow not\ looped.$$

Given the input (i1) with $n = 4, oNum = 4$, (i2) with Policy (A), and (i3) $maxT(3)$ and $time(T), T \in \{0..3\}$, we find a witness plan `rAt(1,1,0). goTo(4,1,0).rAt(4,1,1).goTo(1,1,1).rAt(1,1,2).goTo(4,1,2).rAt(4,1,3).` in an environment instance `pAt(2,3).obsAt(4,3).obsAt(3,3).obsAt(3,2). obsAt(2,2).` which shows that the agent comes back to the same position and decides the same action as before, i.e., it loops.

4.1 Example Policies

Below shows a list of policies to be checked, other than Policy (A):

(B) "move to the farthest unvisited point you observe"

(C) "move to the farthest point you observed until now"

(D) "move to the farthest unvisited point you observed until now"

Appendix

Example definition for *reachable*:

$$\% \text{ compute reachable (obstacle-free) cells from } (1,1)$$
$$reachable(1,2) \leftarrow not\ obsAt(1,2).$$
$$reachable(2,1) \leftarrow not\ obsAt(2,1).$$
$$reachable(X_1,Y) \leftarrow reachable(X,Y), X{-}X_1{=}1, not\ obsAt(X_1,Y).$$
$$reachable(X_1,Y) \leftarrow reachable(X,Y), X_1{-}X{=}1, not\ obsAt(X_1,Y).$$
$$reachable(X,Y_1) \leftarrow reachable(X,Y), Y{-}Y_1{=}1, not\ obsAt(X,Y_1).$$
$$reachable(X,Y_1) \leftarrow reachable(X,Y), Y_1{-}Y{=}1, not\ obsAt(X,Y_1).$$

Example definition for *observed*:

$$observed(X,Y,T) \leftarrow rAt(X,Y,T).$$
$$observed(X_1,Y,T) \leftarrow rAt(X,Y,T), row(X_1), X_1 > X, \#count\{X_2 : obsAt(X_2,Y), row(X_2), X_2 > X, X_2 \leq X_1\} \leq 0.$$
$$observed(X_1,Y,T) \leftarrow rAt(X,Y,T), row(X_1), X_1 < X, \#count\{X_2 : obsAt(X_2,Y), row(X_2), X_2 < X, X_2 \geq X_1\} \leq 0.$$
$$observed(X,Y_1,T) \leftarrow rAt(X,Y,T), column(Y_1), Y_1 > Y, \#count\{Y_2 : obsAt(X,Y_2), column(Y_2), Y_2 > Y, Y_2 \leq Y1\} \leq 0.$$
$$observed(X,Y_1,T) \leftarrow rAt(X,Y,T), column(Y_1), Y_1 < Y, \#count\{Y_2 : obsAt(X,Y_2), column(Y_2), Y_2 < Y, Y_2 \geq Y1\} \leq 0.$$

Example definition for *farthest* w.r.t. Policy (A):

$$
\begin{aligned}
& dist(0..n). \\
distInDir(X,Y,south,D) &\leftarrow rAt(X,Y,T), dist(D), \\
&\quad \#max\{X_1{-}X : observed(X_1,Y,T), row(X_1), X_1{\geq}X\}{=}D. \\
distInDir(X,Y,north,D) &\leftarrow rAt(X,Y,T), dist(D), \\
&\quad \#max\{X{-}X_1 : observed(X_1,Y,T), row(X_1), X{\geq}X_1\}{=}D. \\
distInDir(X,Y,east,D) &\leftarrow rAt(X,Y,T), dist(D), \\
&\quad \#max\{Y_1{-}Y : observed(X,Y_1,T), column(Y_1), Y_1{\geq}Y\}{=}D. \\
distInDir(X,Y,west,D) &\leftarrow rAt(X,Y,T), dist(D), \\
&\quad \#max\{Y{-}Y_1 : observed(X,Y_1,T), column(Y_1), Y{\geq}Y_1\}{=}D. \\
existsMoreThan(X,Y,D) &\leftarrow distInDir(X,Y,P,D), distInDir(X,Y,P_1,D_1), D_1 > D, P_1 \neq P. \\
farthest(X,Y,X{+}D,Y) &\leftarrow distInDir(X,Y,south,D), \mathit{not}\ existsMoreThan(X,Y,D). \\
farthest(X,Y,X{-}D,Y) &\leftarrow distInDir(X,Y,north,D), \mathit{not}\ existsMoreThan(X,Y,D). \\
farthest(X,Y,X,Y{+}D) &\leftarrow distInDir(X,Y,east,D), \mathit{not}\ existsMoreThan(X,Y,D). \\
farthest(X,Y,X,Y{-}D) &\leftarrow distInDir(X,Y,west,D), \mathit{not}\ existsMoreThan(X,Y,D).
\end{aligned}
$$