Inducing a system that intentionally hides

R. Otero, J. Romero, J. Gonzalez and A. Illobre
{otero,jromerod,jgonzalezi,infjdi00}@udc.es

Computer Science Department, University of Corunna (Spain)

Abstract. We study a structure where one dynamic system intentionally hides another dynamic system. We face the problem of discovering the part of the second system that initially is hidden. This study is interesting for Experimental Psychology because it can improve the surveys used for gathering experimental data, where people intentionally hide relevant information. The method we propose consists in applying Inductive Logic Programming for learning an action description of both systems and then using Answer Set Programming with the new theory for planning which actions to execute to discover the hidden parts. We report experiments done with an example of this structure.

1 Introduction

Consider the following problem.

Example 1 (Covered Circle). There is a circle of 13 letters. On top of it there is a cover that hides some of the positions of the circle so that not all the letters can be seen. There are also two tabs. Clicking on a tab makes the letters and the cover change their positions. The problem is: Which are the hidden letters?

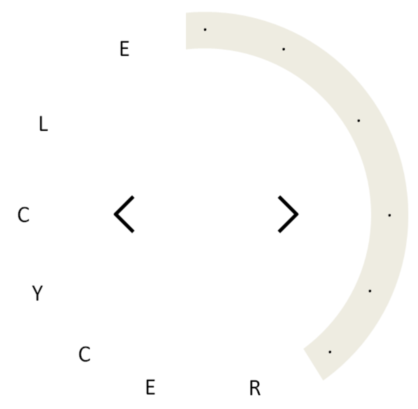

Fig. 1. Covered circle

This is a dynamic system. Letters change their positions in the circle depending on which tab is clicked and on the previous state of the circle. Positions covered depend on the tab and on the previous state of both the circle and the cover. For example, if we click on the left tab the system changes to figure 2. The cover

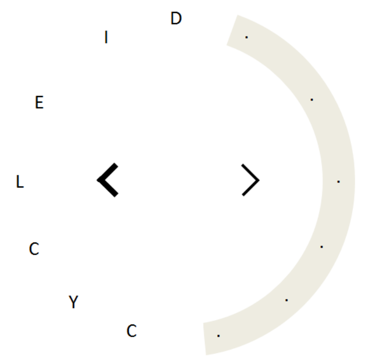

Fig. 2. Covered circle after clicking on the left tab

has moved clockwise while the letters in the circle have moved counterclockwise. As a result now we can see letters ‘I’ and ‘D’. To solve the problem we must continue clicking the tabs until all the remaining letters can be seen.

At first we do not know what are the effects of the actions so we have to click the tabs at random. However, once we have some evidence on the evolution of the domain we can *learn* how does it change and then we can use this knowledge for *planning* which actions to execute. The method we propose combines two logic programming methods for solving these tasks. First we apply Inductive Logic Programming (ILP) [4] to induce an action description of both the circle and the cover and then we apply Answer Set Programming (ASP) [1] for planning with the new action description.

The problem we consider is similar to McCarthy’s Appearance and Reality problem [3] but in this case we already know that there are two layers: a circle and a cover. The circle corresponds to the reality, while the cover and the visible part of the circle correspond to the appearance in McCarthy’s problem.

In this paper we study a structure where one dynamic system intentionally hides another dynamic system. This structure is motivated by our recent work in Experimental Psychology. The objective of this field is building models of human behavior supported by experimental data. Very often this data is collected using surveys that are answered by people, but surveys are not a safe measure instrument because people may intentionally hide some information. For example, when measuring behavior related to environmental issues, a person may say that she behaves very ecologically because she recycles paper at work, while she is hiding that at home she does not recycle at all. Our aim is to apply the new structure to solve this type of problems and improve the quality of the surveys.

For instance, we could model the previous example with a dynamic system that represents the real behavior of the person and another dynamic system that hides part of the real behavior. Then a survey could contain some questions to observe the visible part of the real behavior and other questions to try to change which part of the real behavior is visible. For example, a question could appeal to the honesty of the person or could stress the anonymity of the surveys to decrease the social pressure over the person.

2 Experiments

We report two experiments with the Covered Circle example. We use constants $s0$, $s1$, ... for situations, $p1$, $p2$, ..., $p13$ for positions, r, e, ... for 13 letters of the circle and $left$ and $right$ for actions. Also, the following predicates:

Fluents:

$circle(S, P, L)$: At situation S the letter L is in position P.
$prevcircle(S, P, L)$: At the situation previous to S the letter L is in position P.
$cover(S, P)$: At situation S the position P is covered.
$prevcover(S, P)$: At the situation previous to S the position P is covered.
$nprevcover(S, P)$: At the situation previous to S the position P is not covered.
$prevcoverl(S, L)$: At the situation previous to S the letter L is covered.
$nprevcoverl(S, L)$: At the situation previous to S the letter L is not covered.

Actions:

$do(S, A)$: At situation S action A is executed.

Static predicates:

$next(P_1, P_2)$: Position P_2 goes after (clockwise) position P_1.
$eqp(P_1, P_2)$: Position P_1 is equal to position P_2.
$eql(L_1, L_2)$: Letter L_1 is equal to letter L_2.

2.1 Experiment 1

Step 1. Gathering evidence. In a real-world problem we do not know in advance what are the effects of the available actions. So first we would do some actions to gather evidence of the evolution of the domain, and then we would use that evidence for learning a description of the effects of actions that finally we would use for planning which actions to execute. In our experiments to simulate the behavior of the real system we apply ASP to solve a prediction problem. The problem is represented by a logic program R that contains the action theory A_R for fluents $circle$ and $cover$ and a set of facts I_R about the initial situation.

In experiment 1 both the real circle and the real cover behave as a dial: they move clockwise when clicking $right$ and they move counterclockwise when clicking $left$. A_R is:

$$
\begin{gathered}
\mathit{circle}(S,P,L) \mathrel{:-} \mathit{do}(S,\mathit{right}),\ \mathit{next}(Q,P),\ \mathit{prevcircle}(S,Q,L).\\
\mathit{circle}(S,P,L) \mathrel{:-} \mathit{do}(S,\mathit{left}),\ \mathit{next}(P,Q),\ \mathit{prevcircle}(S,Q,L).\\
\mathit{cover}(S,P) \mathrel{:-} \mathit{do}(S,\mathit{right}),\ \mathit{next}(Q,P),\ \mathit{prevcover}(S,Q).\\
\mathit{cover}(S,P) \mathrel{:-} \mathit{do}(S,\mathit{left}),\ \mathit{next}(P,Q),\ \mathit{prevcover}(S,Q).
\end{gathered}
$$

We include A_R in program R, but this theory is not used in the learning and planning steps to simulate that it is unknown to the learning agent. In experiment 1 the set of facts I_R represent the initial situation of figure 1. We add to R a set of action occurrences representing the execution of actions $left$, $right$ and $right$:

$$\mathit{do}(s1, \mathit{left}).\ \mathit{do}(s2, \mathit{right}).\ \mathit{do}(s3, \mathit{right}).$$

Running the answer set solver *Clasp*[1] with the final program we find the unique answer set of the program, whose atoms represent the evolution of the system. Really R is encoded in such a way that we only get the atoms that represent what the learning agent would observe, i.e., if $\mathit{hidden}(s,p)$ belongs to the answer set then for any letter l, $\mathit{circle}(s,p,l)$ does not belong to it. For this experiment the solution to the prediction problem is represented in the following table:

situation	action	circle
s0		. . . . . . r e c y c l e
s1	left	. . . . . r e c y c l e .
s2	right	. . . . . . r e c y c l e
s3	right	e . . . . . . r e c y c l

Step 2. Learning. We induce an action description for fluents *circle* and *cover* given the evidence of the previous step. We apply the method of *Inverse Entailment* (*IE*) [4] implemented in the *ILP* system *Progol* [4].

The input to *Progol* consists of the *modes* definitions, the previous answer set represented as a list of facts, and one constraint for each target fluent to guarantee the consistency of the final theory. The solution found by *Progol*, A_H, is equivalent to A_R.

Step 3. Planning. We apply ASP [2] to solve a planning problem using the action theory A_H found in the previous step. Initial situation is $s3$ and the goal is to discover all the hidden letters[2]. In this experiment the planning problem has no solution and *Clasp* returns no answer set. The cover moves in the same direction as the circle, so there is no chance to uncover any of the hidden letters: the problem cannot be solved. Note, however, that in the real-world example we do not know that A_H is equivalent to A_R, so it would make sense to acquire more evidence to try to improve A_H and solve the problem.

[1] `http://www.cs.uni-potsdam.de/clasp/`

[2] We add facts about the hidden letters (l_1 to l_6) at $s3$ ($\mathit{circle}(s3, p2, l_1)$. $\ldots$ $\mathit{circle}(s3, p7, l_6)$.) and state the goal that for each letter l_i exists a situation s and a position p such that $\mathit{circle}(s,p,l_i)$ holds and $\mathit{hidden}(s,p)$ does not hold.

2.2 Experiment 2

Step 1. Gathering evidence. In this experiment the real system is different. Thus the real action theory A_R, which we will see later, is also different. We solve with ASP the same prediction problem as in the previous experiment using the new action theory, and this is the solution:

situation	action	circle
s0		. . . . . . r e c y c l e
s1	left	. . . . . . e c y c l e **i**
s2	right	. . . . . . r e c y c l e
s3	right	. . . . . . **t** r e c y c l

Step 2. Learning. Given the previous evidence *Progol* finds the following action theory A_H, the circle moves as a dial and the cover persists[3] :

$$circle(S,P,L) \text{:-}\ do(S,right),\ next(Q,P),\ prevcircle(S,Q,L).$$
$$circle(S,P,L) \text{:-}\ do(S,left),\ next(P,Q),\ prevcircle(S,Q,L).$$
$$cover(S,P) \text{:-}\ prevcover(S,P).$$

Step 3. Planning. We apply ASP to solve a planning problem with A_H. The shortest plan found by *Clasp* has four situations:

$$do(s4, right).\ do(s5, right).\ do(s6, right).\ do(s7, right).$$

According to A_H the cover persists, so moving the circle clockwise we would see all the remaining letters.

Step 4. Testing. In a real-world problem we would execute the previous plan in the real system. Instead, we apply ASP to solve a prediction problem with that plan and action theory A_R. This is the solution:

situation	action	circle
s4	right	l . . . . . . t r e c y c
s5	right	c l . . . . . . t r e c y
s6	right	y c l . . . . . . t r e c
s7	right	c y c l . . . . . . t r e

The plan was correct for a system that behaved according to A_H, but the results show that A_H is not a correct action description of the real system.

Step 5. Learning. We repeat the learning step, this time using the evidence of situations $s0$ to $s7$. *Progol* finds the following action description A'_H:

$$circle(S,P,L) \text{:-}\ do(S,right),\ next(Q,P),\ prevcircle(S,Q,L).$$
$$circle(S,P,L) \text{:-}\ do(S,left),\ next(P,Q),\ prevcircle(S,Q,L).$$
$$cover(S,P) \text{:-}\ next(Q,P),\ prevcover(S,Q),\ nprevcoverl(S,t).$$
$$cover(S,P) \text{:-}\ prevcover(S,P),\ prevcoverl(S,t).$$

The circle behaves as a dial, and the cover persists if letter 't' is covered, but moves clockwise in other case.

[3] Last rule is a "frame axiom", to solve the frame problem in induction and avoid the induction of this type of rules we propose to use the method of [5].

Step 6. Planning. With A'_H the shortest plan found by *Clasp* has five situations:

$$do(s8, left).\ do(s9, left).\ do(s10, left).\ do(s11, left).\ do(s12, left).$$

According to A'_H, if we move the circle counterclockwise then the letter 't' will be covered, the cover will stop moving and we will uncover the remaining letters.

Step 7. Testing. Action theory A'_H is equivalent to A_R and the plan is correct for the real system. This is the solution of the prediction problem with A_R:

situation	action	circle
s8	left	y c l e i . . . . . . e c
s9	left	c l e i **d** . . . . . . c y
s10	left	l e i d **o** . . . . . . y c
s11	left	e i d o **n** . . . . . . c l
s12	left	i d o n **o** . . . . . . l e

Finally there was a message hidden in the circle: "I do not recycle".

3 Conclusions and Future Work

We have studied a structure where one dynamic system intentionally hides another dynamic system. We report two experiments. In the first one *Progol* induced the real action description of both systems, but it was not possible to uncover the hidden parts. In the second experiment it was necessary to repeat the learning step adding more evidence on the domain, but then *Progol* induced the real action description, that finally was used with *Clasp* to find a plan that uncovered the hidden parts of the second system.

This approach can be extended studying other examples of the structure, for instance, systems that grow or shrink over time. Also, the properties of the structure can be formally studied. For example, it would be interesting to determine the conditions under which the action descriptions can be induced and the hidden parts can be uncovered. Finally, we plan to apply the results of this line of work to the design of a system for online surveys that dynamically chooses the questions to be asked to the people.

References

1. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
2. V. Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138:39–54, 2002.
3. J. Mccarthy. Appearance and reality: a challenge to machine learning. http://www-formal.stanford.edu/jmc/appearance.html, 1999.
4. S. H. Muggleton. Inverse entailment and progol. *New Generation Computing*, 13:245–286, 1995.
5. R. Otero. Induction of the effects of actions by monotonic methods. *Proc. of the 13th Int. Conf. on Inductive Logic Programming, ILP-03. LNAI*, 2835:193–205, 2003.