Learning Relational Grammars from Sequences of Actions

Blanca Vargas-Govea and Eduardo F. Morales

National Institute of Astrophysics, Optics and Electronics
Computer Science Department
Luis Enrique Erro 1, 72840 Tonantzintla, México
{blanca,emorales}@inaoep.mx

Abstract. Many tasks can be described by sequences of actions that normally exhibit some form of structure and that can be represented by a grammar. This paper introduces FOSeq, an algorithm that learns grammars from sequences of actions. The sequences are given as low-level traces of readings from sensors that are transformed into a relational representation. Given a transformed sequence, FOSeq identifies frequent sub-sequences of n-items, or n-grams, to generate new grammar rules until no more frequent n-grams can be found. From m sequences of the same task, FOSeq generates m grammars and performs a generalization process over the best grammar to cover most of the sequences. The grammars induced by FOSeq can be used to perform a particular task and to classify new sequences. FOSeq was tested on robot navigation tasks and on gesture recognition with competitive performance against other approaches based on Hidden Markov Models.

1 Introduction

Sequences are used to describe different problems in many fields, such as natural language processing, music, DNA, and gesture recognition, among others. The analysis of such sequences often involves exploiting the information provided by the implicit structure of the sequences that can sometimes be represented by a grammar. Learning grammars from sequences offers several advantages. Suppose that you have a set of sequences of actions performed by a robot to move between two designated places avoiding obstacles. If we could infer a grammar from such sequences, we could use it to recognize when a robot moves between two places and also to generate a sequence of actions to perform such task. Another advantage is that a grammar normally includes sub-concepts that can include other sub-concepts or primitive actions, which can be used to solve other related sub-tasks.

Grammars can be represented by different formalisms, the most commonly used is context-free grammars (CFGs). In this paper, we use Definite Clause Grammars (DCGs), a generalization of CFGs that use a relational representation. This is important as it allows us to apply the learned grammar to different instantiations of a more general problem.

We focus on learning grammars from sequences of actions that can be used as programs to execute a task and as classifiers. The training sequences are provided by the user, the main idea is to show the system what to do instead of how to do it (e.g., steering the robot avoiding obstacles or moving a hand), simplifying the programming effort. The set of traces consists of low-level sensor readings that are transformed into a high level representation. The transformed sequences are given to an algorithm (FOSeq) that induces grammars that can be used to reproduce the original human-guided traces and to identify new sequences.

The approach was applied in two domains: (i) robot navigation and (ii) gesture recognition. We tested the learned navigation grammars in a robotics scenario with both simulated and real environments and show that the robot is able to accomplish several navigation tasks. The gesture recognition was tested using a public database of gestures, showing that the classification accuracy is competitive with other common approaches with the advantage of learning an understandable representation.

This paper is organized as follows. Section 2 reviews related work. Section 3 describes the grammar learning algorithm. Section 4 presents the navigation task while Section 5 describes the gesture recognition experiment. Conclusions and future research directions are given in Section 6.

2 Related Work

Grammar induction has been commonly studied in the context of Natural Language Processing. Most of the approaches use grammars as parsers and focus on specific problems and are rarely used to solve other related problems. Other researchers have tried to induce grammars using a relational representation. EMILE [1] and ABL [2] are algorithms based on first order logic that learn the grammatical structure of a language from sentences. Both algorithms focus on language and it is not easy to extend them to other applications. GIFT [3] is an algorithm that learns logic programs but it requires an initial rule set given by an expert. In contrast, FOSeq learns relational grammars from sequences of actions, where the grammars are logic programs that can reproduce the task described by the sequences. A grammar induction technique closely related to our work is SEQUITUR [4], an algorithm that infers a hierarchical structure from a sequence of discrete symbols. However, SEQUITUR handles only constant symbols, is based on bi-grams (sets of two consecutive symbols), and consequently, the learned rules can only have pairs of literals in their bodies, and it does not generalize rules. FOSeq employs a relational approach, is not restricted to bi-grams and is able to generalize.

Learning from sequences has been used in robotics to learn skills. In [5] a robot has to learn movements (e.g., aerobic-style movement) showed by a teacher. To encode the movements, and subsequently be able to recognize them, they used Hidden Markov Models (HMM). However, this representation is not easy to interpret and does not capture the hierarchical structure of the sequences. In our approach, hierarchical skills can be learned from sequences of basic skills.

Another domain that has been used to learn from sequences is gesture recognition, which is an important skill for human–computer interaction. Hidden Markov Models and neural networks are standard techniques for gesture recognition. However, most of the approaches have emphasized on improving learning and recognition performance without considering the understandability of the representation [6].

3 Learning Grammars with FOSeq

The general algorithm can be stated as follows: from a set of sequences, (i) learn a grammar for each sequence, (ii) parse all the sequences with each induced grammar, evaluate how well each grammar parses all the traces, and (iii) apply a generalization process to the best grammar trying to cover most of the sequences.

1: Grammar induction. Given a trace of predicates the algorithm looks for n-grams (e.g., sub-sequences of n-items, in our case n-predicates) that appear at least twice in the sequence. As in Apriori [7], the candidate n-grams are incrementally searched by their length. The search starts with $n = 2$ and ends when there are no more repeated n-grams for $n \geq 2$. The n-gram with the highest frequency of each iteration is selected, generating a new grammar rule and replacing in the sequence, all occurrences of the n-gram with a new non-terminal symbol. If there is more than one n-gram with the same highest-frequency, the longest n-gram is selected. If there are several candidates of the same length, the algorithm randomly selects one of them. Repeated items are removed because items represent actions that are executed continuously while specific conditions hold. Therefore, the action will be repeated while its conditions are satisfied even if it appears only once.

Example. Let us illustrate the grammar induction process with the following sequence of constants: S $\rightarrow$ a b c b c b c b a b c d b e b c. FOSeq looks for n-grams with frequency ≥ 2 as candidates to build a rule. In the first iteration there is only one candidate: {b c} with five appearances in the sequence. This n-gram becomes the body of the new rule R1$\rightarrow$b c. The n-gram is replaced by the non-terminal R1 in sequence S, generating S_1 and removing repeated items. In the second iteration FOSeq finds three candidates: {R1 b}, {a R1} and {a R1 b} with two repetitions each. FOSeq selects the longest item: {a R1 b} and a new rule is added: R2$\rightarrow$a R1 b. Sequence S_2 does not have repeated items and the process ends. Figure 1 shows the learned grammar where R1 and R2 can be seen as sub-concepts in the sequence.

When the items of the sequence are first–order predicates the learned grammar is a definite clause grammar (DCG). DCGs are an extension of context free grammars that are expressed and executed in Prolog. In this paper we have sequences of predicates that are state-action pairs with the following format: pred1($State_1$,$action_1$), pred2($State_2$,$action_1$), pred1($State_3$,$action_2$), ...

For repeated predicates a new predicate is created, where *State* is the first state of the first predicate and *Action* is the action of the last predicate. For

$S_2 \rightarrow$ R2 d b e R1
R1 $\rightarrow$ b c
R2 $\rightarrow$ a R1 b

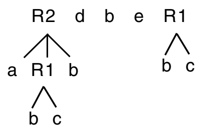

Fig. 1. Induced grammar for sequence S. S_2 is the compressed sequence after the induction process. The grammar describes the hierarchical structure of the sequence.

instance, suppose that the following pair of predicates is repeated several times in the sequence: $\ldots$, pred1($State_j$,$Action_1$), pred2($State_k$,$Action_2$), $\ldots$, then the following new predicate is created:
newpred($State_1$,$Action_2$) $\leftarrow$ pred1($State_1$,$Action_1$), pred2($State_2$,$Action_2$) .

2: Grammar evaluation. A grammar is created for each sequence. Every learned grammar is used to parse all the sequences in the set of traces provided by the user and the best evaluated grammar is selected. The measure of how well the grammar parses is calculated using the following function:

$$eval(g_i) = \sum_{i=1}^{m} \frac{c_i}{c_i + f_i}$$

where g_i is the grammar being evaluated from a set of m sequences, c_i and f_i are the number of items that the grammar is able or unable to parse respectively and i is the index of the sequence being evaluated. When a grammar is not able to parse a symbol, it is skipped. FOSeq selects the grammar that best parses the set of sequences.

3: Generalization. The key idea of the generalization process is to obtain a new grammar that improves the covering of the best grammar. It is performed using pairs of grammars. For example, if the best grammar describes the trajectory of a mobile robot when its goal is to the right, we would expect that another sequence provides information about how to reach a goal to the left of the robot. The generalization process generates a clause that covers both cases calculating the *lgg* (least general generalization [8]) of both clauses. The process can be summarized as follows:

1. Select the best grammar g_{best}
2. Select the grammar g_{other} that provides the largest number of different instantiations of predicates.
3. Compute the *lgg* between grammar rules of g_{best} and g_{other} with different instantiations of predicates and replace the grammar rule from g_{best} by the resulting generalized rule.
4. If the new grammar rule improves the original coverage, it is accepted, otherwise it is discarded.
5. The process continues until a coverage threshold is reached, g_{best} rules cover all the rules in the other grammars or there is no longer improvement with the generalization process.

Table 1. lgg example

c_1	c_2	lgg(c_1,c_2)
pred($State$,$action_2$) $\leftarrow$ cond1($State$,$action_1$), cond2($State$,$action_2$).	pred($State$,$action_3$) $\leftarrow$ cond1($State$,$action_1$), cond2($State$,$action_3$).	pred($State$,$Action$) $\leftarrow$ cond1($State$,$action_1$), cond2($State$,$Action$).

The generalization process is used to produce a more general grammar applicable to different traces of the problem. Table 1 shows the *lgg* of clauses c_1 and c_2 where the constants $action_2$ and $action_3$ are replaced by the variable $Action$.

4 Learning Navigation Tasks

When a mobile robot navigates through an office/house environment it describes a trajectory that can be represented by sequences of actions. Our approach is based on a teleo-reactive framework where learned DCGs represent Teleo-Reactive Programs (TRPs) [9]. TRPs are sets of reactive rules that sense the environment continuously and apply an *action* whose continuous execution eventually satisfies a goal condition. The following basic navigation TRPs are learned in [10]: *wander*, *orient*, *leave-a-trap* and *follow-a-mobile-object*.

Learning to go to a target point. Given a sequence consisting of *wander* and *orient* FOSeq is used to learn a grammar that can go between two places using such skills and possibly inducing intermediate concepts. The user steered the mobile robot with a joystick to reach different goals producing 8 traces. FOSeq learned 8 grammars, one for each trace. After being evaluated, the best induced grammar covered 99.29% of the traces. Table 2 shows the generalized rules learned from the trace of 8 sequences. Predicate names were given by the user. Each rule describes a sub-task along the navigation trajectory. For example, R1 describes the "turning-to-goal" behavior and R2 describes the "direct-to-goal" behavior when the robot does not need to turn because the goal is in its same direction and it wanders to reach the goal. Table 3 shows other hierarchical TRPs learned using other basic skills: *wander*, *orient*, *follow* and *leave-trap*.

Navigation experiments. The experiments were carried out in simulation and with a service ActivMedia robot equipped with a sonar ring and a Laser SICK LMS200 using the Player/Stage software [11]. The goal of the experiments is to show that the learned TRPs can control the robot in completely unknown and dynamic environments. We evaluate the performance of the learned TRPs in 10 different scenarios, with different obstacles' sizes and shapes, and with static and dynamic obstacles. The TRPs were evaluated by the percentage of tasks successfully completed, and the number of operator interventions (e.g., if the robot enters a loop). The robot's initial position and the goal point (when applicable) were randomly generated. In each experiment two operator interventions were allowed, otherwise, the experiment failed. Table 3 summarizes the results

Table 2. Goto TRP rules

(R1) turning-to-goal($State_1$,*go-fwd*)	$\rightarrow$ orient($State_1$,*Action*), wander($State_2$,*go-fwd*)
(R2) direct-to-goal($State_1$,*Action*)	$\rightarrow$ orient($State_1$,*equal*), wander($State_2$,*Action*)
(R3) cannot-orient($State_1$,*Action*)	$\rightarrow$ orient($State_1$,none),wander($State_2$,*Action*)
(R4) ramble(*State*,*Action*)	$\rightarrow$ wander(*State*,*Action*)

Table 3. Accuracy: Hierarchical TRPs

TRP	#seq.	Tasks	Int.	Acc1	Acc2
wander + orient (goto)	8	30	2	93.33	86.67
wander + orient + leave-trap	12	30	1	96.67	93.33
follow + wander	10	40	2	95	90
follow + wander + leave-trap	14	40	0	100	100

in simulation. It is shown the number of tasks to test each TRP, the operator interventions and the accuracy with (Acc1) and without interventions (Acc2).

The learned TRPs were integrated as the navigation module of a PeopleBot service robot [12]. The given tasks were: (i) following a human under user commands, (ii) navigating to several places in the environment. Each place has a previously defined name (e.g., kitchen, sofa), (iii) finding one of a set of different objects in a house, and (iv) delivering messages and/or objects between different people. The first three tasks are part of the RoboCup@Home challenge. Navigation and follow videos can be seen at: http://www.youtube.com/user/anon9899

5 Dynamic Gesture Recognition

Interacting with gestures is a natural human ability that can improve the human-computer communication. In this section it is described how to learn grammars from sequences of dynamic gestures and use them to classify new sequences.

FOSeq transforms low-level information from sensors into a relational representation. We have sequences of state-value pairs, where a value can be an action or a boolean variable, as described below. We used a database[1] of 7308 samples from 9 dynamic gestures taken from 10 men and 5 women. Figure 2(a) shows the initial and final position for each gesture. The whole set can be seen in Figures 2(b)-(j). Gestures were executed with the right arm and they were obtained using the monocular visual system described in [6].

Each sequence is a vector with sets of seven attributes describing the executed gesture. An example of a sequence is: $(+ + - -$ T F F$)$,$(+ + - +$ T F F$)$, … where the first three attributes of each vector describe motion and the rest describe posture. Motion features are $\Delta area$, Δx and Δy, representing changes in hand area and changes in hand position of the XY-axis of the image respectively. These features take one of three possible values: $\{+,-,0\}$ indicating increment,

[1] Database available at http://sourceforge.net/projects/visualgestures/

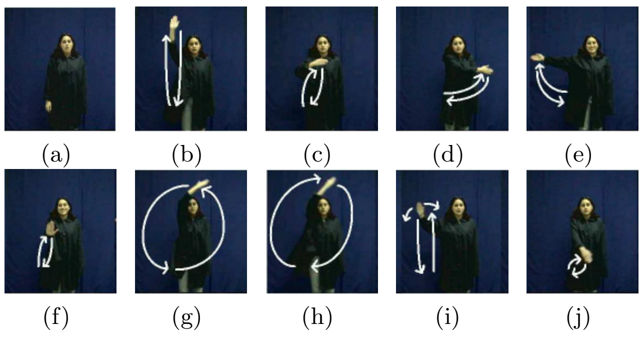

Fig. 2. Gesture set: (a) initial and final position, (b) attention, (c) come, (d) left, (e) right, (f) stop, (g) turn-right, (h) turn-left, (i) waving, (j) pointing

decrement or no change, according to the area and position of the hand in a previous image. Posture features are: *form, above, right* and *torso*, and describe hand appearance and spatial relations between the hand and face/torso. Hand appearance is described by *form*. Possible values for this feature are {+,−,0}: (+) if the hand is vertical, (−) if the hand has horizontal position, and (0) if the hand is tilted to the right or left over the XY plane. Features *right* and *torso* indicate if the hand is to the right of the head and over the torso. Based on this information, sequences are transformed into a first-order representation by replacing their attributes with a meaningful fact (e.g., hmove($State$, $right$), vmove($State$, up),size($State$,inc), shape($State$, $vertical$), ...).

We focused on the recognition between gestures produced by one person following the experimentation setting described in [6]: (i) from 50 sequences of each gesture, randomly select 20 sequences to learn the grammars and build training sets of 2 and 10 sequences, (ii) learn a grammar for each gesture, (iii) test the grammars with the remaining 30 sequences, (iv) repeat 10 times.

The overall accuracy obtained by FOSeq and the HMM approach in [6] is as follows: both training sub-sets FOSeq performs similar to HMMs: with 2 training sequences, FOSeq got 95.17%, whereas HMMs 94.85%. With 10 training sequences, FOSeq got 97.34%, and HMMs 97.56%. These results are very promising as HMM is the leading technique in this application. Table 4 shows the confusion matrix that expresses the proportion of true classified instances for 10 training sequences. The best classified gestures are: *left, right, turn-right* and *waving* (100%) whereas *pointing* is the worst classified gesture (85.17%). Misclassification are concentrated between *pointing* and *come.*

Learned grammars for *pointing* and *come* have 5 common rules whereas *right* and *left* grammars do not have any. For instance, an identical grammar rule for *pointing*/*come* is: R1 → above_face($State$,$false$) over_torso($State$,$true$) explaining that the hand is not near the face but it is over the torso. This type of similarities and the identification of common sub–gestures is not possible to obtain with other approaches. Learning relational grammars for gesture recognition produces an explicit representation of rules and is able to identify and

Table 4. Confusion matrix for 10 sequences. Classes: 1) *attention*, 2) *come*, 3) *left*, 4) *pointing*, 5) *right*, 6) *stop*, 7) *turn-left*, 8) *turn-right*, 9) *waving*

	1	2	3	4	5	6	7	8	9	
1	**99.33**							0.67		300
2		**97.24**		2.76						290
3			**100**							290
4		13.10	1.72	**85.17**						290
5					**100**					300
6	1.03					**97.59**			1.38	290
7					2.07		**96.55**		1.38	290
8								**100**		290
9									**100**	290
	301	320	295	255	306	283	280	292	298	2630

generate rules for sub-gestures. It also helps to find similarities between different gestures and has a competitive performance against HMMs.

6 Conclusions and Future Work

In this work we have introduced an algorithm called FOSeq, that takes sequences of states and actions and induces a grammar able to parse and reproduce the sequences. FOSeq learns a grammar for each sequence, followed by a generalization process between the best evaluated grammar and other grammars to produce a generalized grammar covering most of the sequences. Once a grammar is learned it is transformed into a TRP in order to execute particular actions and achieve a goal. FOSeq was able to learn a navigation grammar from sequences given by the user and used the corresponding TRP to guide the robot in several navigation tasks in dynamic and unknown environments. FOSeq was also used to learn grammars from gesture sequences with very competitive results when compared with a recent state-of-the-art system. As part of our future work, we are working on learning more TRPs to solve other robotic tasks, we plan to extend the experiments with gestures to obtain a general grammar for a gesture performed by more than one person, and we are interested in reproducing gestures with a manipulator.

The authors thank the reviewers for their useful comments. This research was sponsored by CONACYT under grant 203873 .

References

1. Adriaans, P.W., Trautwein, M., Vervoort, M.: Towards high speed grammar induction on large text corpora. In: Jeffery, K., Hlaváč, V., Wiedermann, J. (eds.) SOFSEM 2000. LNCS, vol. 1963, pp. 173–186. Springer, Heidelberg (2000)
2. van Zaanen, M.V.: Abl: alignment-based learning. In: Proc. 17th Conf. on Computational linguistics, Association for Computational Linguistics, pp. 961–967 (2000)

3. Bernard, M., de la Higuera, C.: Gift: Grammatical inference for terms. In: International Conference on Inductive Logic Programming (1999)
4. Nevill-Manning, C., Witten, I.: Identifying hierarchical structure in sequences: A linear-time algorithm. Journal of Artificial Intelligence Research 7, 67–82 (1997)
5. Amit, R., Mataric, M.J.: Learning movement sequences from demonstration. In: ICDL 2002: Proc. 2nd International Conf. on Development and Learning, Cambridge, MA, pp. 12–15 (2002)
6. Avilés-Arriaga, H., Sucar, L., Mendoza, C.: Visual recognition of similar gestures. In: 18 International Conference on Pattern Recognition ICPR 2006, vol. 1, pp. 1100–1103 (2006)
7. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proc. 20th Int. Conf. Very Large Data Bases, VLDB, pp. 487–499. Morgan Kaufmann, San Francisco (1994)
8. Plotkin, G.: A note on inductive generalization. Machine Intelligence 5, 153–163
9. Benson, S., Nilsson, N.J.: Reacting, planning, and learning in an autonomous agent. Machine Intelligence 14, 29–62 (1995)
10. Vargas, B., Morales, E.F.: Learning navigation teleo-reactive programs using behavioural cloning. In: IEEE International Conference on Mechatronics (2009)
11. Vaughan, R.T., Gerkey, B.P., Howard, A.: On device abstractions for portable, reusable robot code. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2421–2427 (2003)
12. Avilés, H.H., Sucar, L.E., Morales, E.F., Vargas, B.A., Sánchez, J., Corona, E.: Markovito: A flexible and general service robot, January 2009. Studies in Computational Intelligence, vol. 177, pp. 401–423. Springer, Heidelberg (2009)