Learning Static Constraints for Domain Modeling from Training Plans

Rabia Jilani

University of Huddersfield, Huddersfield UK
rabia.jilani@hud.ac.uk

Abstract. Intelligent agents solving problems in the real-world require domain models containing widespread knowledge of the world. Synthesising operator descriptions and domain specific constraints by hand for AI planning domain models is time-intense, error-prone and challenging. To alleviate this, automatic domain model acquisition techniques have been introduced. For example, the LOCM system requires as input some plan traces only, and is effectively able to automatically encode the dynamic part of the domain model. However, the static part of the domain, i.e., the underlying structure of the domain that can not be dynamically changed, but that affects the way in which actions can be performed is usually missed, since it can hardly be derived by observing transitions only.
In this paper we introduce ASCoL, a tool that exploits graph analysis for automatically identifying static relations, in order to enhance planning domain models. ASCoL has been evaluated on domain models generated by LOCM for the international planning competition, and has been shown to be effective.

Keywords: Automated Planning, Knowledge Engineering, Domain Model Acquisition, Domain Constraints.

1 Introduction

Intelligent agents that solve problems in the real-world use models containing vast knowledge of that world to plan their actions. Designing complete domain models and gathering application knowledge is crucial not only for the efficiency of intelligent planning systems, but also for the correctness of the resulting action plans generated by intelligent agents. Creating such models is a difficult task, that is usually done manually; it requires planning experts and it is time-consuming and error-prone.

In this paper we present ASCoL, (Automated Static Constraint Learner), a tool that can effectively identify static knowledge by considering plan traces as the only source of information.

In the following sections, we first provide a general background of the areas and concepts included in this research which is followed by the research question, problem, contribution, the developed system and system evaluation. The paper is concluded with a discussion of some future goals.

2 Background

2.1 AI Planning

The term Planning refers to the process of reasoning about actions and then organizing those actions to accomplish a desired goal. It gives a full scheme for doing or accomplishing something. In Artificial Intelligence (AI) terms this scheme is known as a Plan. Plan generation is considered an important property of intelligent behaviour. AI Planning has been a part of study in artificial intelligence for over two decades.

Classical planning [2] deals with finding a (partially or totally ordered) sequence of actions transforming the static, deterministic and fully observable environment from some initial state to a desired goal state. In the classical representation *atoms* are predicates. *States* are defined as sets of ground (positive) atoms. A *planning operator* $op = (name(o), pre(o), eff^-(o), eff^+(o))$ is specified such that $name(o) = op\_name(x_1, \ldots, x_k)$ (op_name is a unique operator name and $x_1, \ldots x_k$ are variable symbols (arguments of certain types) appearing in the operator), $pre(o)$ is a set of predicates representing the operator's preconditions, $eff^-(o)$ and $eff^+(o)$ are sets of predicates representing the operator's negative and positive effects. *Actions* are ground instances of planning operators. An action $A = (pre(A), eff^-(A), eff^+(A))$ is *applicable* in a state s if and only if $pre(A) \subseteq s$. Application of A in s (if possible) results in a state $(s \setminus eff^-(A)) \cup eff^+(A)$.

2.2 Knowledge Engineering for AI Planning

Knowledge Acquisition and Knowledge Engineering (KE) have been significant to Artificial Intelligence (AI) research since the fields inception. Acquiring domain knowledge from input training data has attracted much interest in research.

The domain model acquisition problem has mainly been tackled by exploiting two approaches. On the one hand, knowledge engineering tools for planning have been introduced over time, for supporting human experts in modelling the knowledge. Two particular examples are itSIMPLE [10] and PDDL studio [8]. Recently, also crowd-sourcing has been exploited for acquiring planning domain models [13]. On the other hand, a number of techniques are currently available for automatic domain model acquisition; they rely on example data for deriving domain models. For example ARMS [12], SLAF [9] and many more. Significant differences can be found in terms of the quantity and quality of the required inputs. [6] presents a brief overview of nine different knowledge engineering tools used in planning and compares these systems on a set of criteria.

2.3 The LOCM System

One such knowledge acquisition tool is LOCM (Learning Object-Centred Models) [1] that carries out automated generation of a planning domain model from a set of example training plans. Each plan is a sequence of actions, where each

action in the plan is stated as a name and a list of objects that are affected or are needed during the actions execution. Different plan traces are generated by observing the behaviour of an agent in its environment.

The LOCM method is unique in that it requires no prior knowledge and can learn action schema without requiring any additional information about predicates or initial, goal or intermediate state descriptions of objects for the example action sequences. The exception to this is that LOCM effectively determines the dynamic part of the domain model of objects but not the static part i.e., the underlying structure of the domain that can not be dynamically changed, but that affects the way in which actions can be performed.

1. Dynamic Knowledge: a set of parametrised operator schema representing generic actions and resulting changes in the domain under study.
2. Static Knowledge: relationships/constraints that are implicit in the set of operators and are not directly expressed but essentially are present while defining a domain model. These can be seen to appear in the preconditions of operators only and not in the effects. In simple words static facts never change in the world. According to Wickler [11], let $O = \{O_1, O_2, \ldots, O_n\}$ be a set of operators and let $Pr = \{Pr_1, Pr_2, \ldots, Pr_n\}$ be a set of all the predicate symbols that occur in these operators. A predicate $Pr_i \in Pr$ is fluent *iff* there is an operator $O_j \in O$ that has an effect that changes the truth of the predicate Pr_i. Otherwise the predicate is static.

In many domains, there are static relationships or constraints which restrict the values of variables in domain modelling. For example learning the map of roads in transport domains or the fixed card stacking rules between specific cards in card-games domains, that never change with the execution of actions.

3 The Research

Our work is aimed at automating the acquisition of static constraints. We aim to enhance the LOCM system to learn complete domain models including the knowledge of static constraints by using sequences of plans as the only input training data. The static knowledge is not explicitly captured in the plan traces and so it is a big challenge to learn such static constraints from them.

We introduced ASCoL, an efficient and effective tool for identifying static knowledge missed by domain models automatically acquired.

The proposed approach generates a directed graph for each pair of same-type arguments of operators and, by analysing linearity properties of the graphs, identifies relevant relations between arguments. Remarkably, the contributions of ASCoL, as demonstrated by our large experimental analysis, are: (i) the ability to identify different types of static relations, by exploiting graph analysis; (ii) ASCoL can work with both optimal and suboptimal plan traces; (iii) considering pairs of same-typed objects allows the identification of all the static relations considered in the benchmark models, and (iv) it can be a useful debugging tool

for improving existing models, which can indicate hidden static relations helpful for pruning the search space.

A preliminary version of ASCoL has been presented in [5]; this version was able to identify inequality constraints only. After further development and a large experimental analysis, in the recent AI*IA publication [7], we demonstrate the ability of ASCoL in finding static relations for enhancing domain models automatically acquired by LOCM.

3.1 The Learning Problem

We define the learning problem that ASCoL addresses as follows. Given the knowledge about object types, operators and predicates, and a set of plan traces, how can we automatically identify the static relation predicates that are needed by operators' preconditions? We base our methodology on the assumption that plan traces contain tacit knowledge about constraints validation/acquisition.

Specifically, a *learning problem description* is a tuple *(P, T)*, where P is a set of plan traces and T is a set of types of action arguments in P (taken from the LOCM learnt domain model). The *output* for a learning problem is a *constraint repository* R that stores all admissible constraints on the arguments of each action A in plan traces P.

4 The ASCoL Tool

We now briefly present the ASCoL tool that has been developed for identifying useful static relations. The process steps can be summarised as follows:

1. Read the partial domain model (induced by LOCM) and the plan traces.
2. Identify, for all operators, all the pairs of arguments involving the same object types.
3. For each of the pairs, generate a directed graph by considering the objects involved in the matching actions from the plan traces.
4. Analyse the directed graphs for linearity and extract hidden static relations between arguments.
5. Run inequality check.
6. Return the extended domain model that includes the identified static relations.

The main information available for ASCoL comes from the input plan traces. As a first control, we remove from the plan traces all the actions that refer to operators that do not contain at least two arguments of the same type.

Even though, theoretically, static relations can hold between objects of different types, they mostly arise between same-typed objects. This is the case in transport domains, where static relations define connections between locations. Moreover, considering only same-typed object pairs can reduce the computational time required for identifying relations. It is also worth noting that, in

Table 1. Overall results on considered domains. For each original domain, the number of operators (# Operators), and the total number of static relations (# SR) are presented. ASCoL results are shown in terms of the number of identified static relationships (Learnt SR) and number of additional static relations provided (Additional SR) that were not included in the original domain model. The seventh and eighth columns indicate respectively the number of plans provided in input to ASCoL, that allows it to converge, and the average number of actions per plan (A/P). The last column shows the CPU-time in milliseconds

Domain	# Operators	# SR	Learnt SR	Add. SR	# Plans	Avg. A/P	CPU-time
TPP	4	7	7	0	7	28	171
Zenotravel	5	4	6	2	4	24	109
Miconic	4	2	2	0	1	177	143
Storage	5	5	5	0	24	15	175
Freecell	10	19	13	0	20	60	320
Hanoi	1	0	1	1	1	60	140
Logistics	6	0	1	1	3	12	98
Driverlog	6	2	2	0	3	12	35
Mprime	4	7	7	0	10	30	190
Spanner	3	1	1	0	1	8	144
Gripper	3	0	1	1	1	14	10
Ferry	3	1	2	1	1	18	130
Barman	12	3	3	0	1	150	158
Gold-miner	7	3	1	0	13	20	128
Trucks	4	3	3	0	6	25	158

most of the cases where static relations involve objects of different types, this is due to a non-optimal modelling process. Furthermore, such relations can be easily identified by naively checking the objects involved in actions; whenever some objects of different type always appear together, they are likely to be statically related.

5 Experimental Evaluation

Remarkable results have been achieved in complex domains, with regards to the number of static relations. We considered fifteen different domain models, taken either from IPCs[1] or from the FF domain collection (FFd)[2].

We selected domains that are encoded using different modelling strategies, and their operators include more than one argument per object type. Table 1 shows the results of the experimental analysis. A detailed interpretation of results can be found in the recent AI*IA publication. All domains but Gripper, Logistics and Hanoi, exploit static relations. Input plans of these domains have been generated by using the Metric-FF planner [4] on randomly generated problems, sharing the same objects. ASCoL has been implemented in Java, and run on a Core 2 Duo/8GB processor. CPU-time usage of the ASCoL is in the range of 35-320 (ms) for each domain.

Interestingly, we observe that ASCoL is usually able to identify all the static relations of the considered domains. Moreover, in some domains it is providing

[1] http://ipc.icaps-conference.org/

[2] https://fai.cs.uni-saarland.de/hoffmann/ff-domains.html

additional static relations, which are not included in the original domain model. Remarkably, such additional relations do not reduce the solvability of problems, but reduce the size of the search space by pruning useless instantiations of operators.

6 Conclusion and Future Goals

We are considering several paths for future work. Grant, in [3], discusses the limitations of using plan traces as the source of input information. ASCoL faces similar difficulties as the only input source to verify constraints are sequences of plans. We are also interested in extending our approach for considering static relations that involve more than two arguments In particular, we aim to extend the approach for merging graphs of different couples of arguments. Finally, we plan to identify heuristics for extracting useful information also from acyclic graphs.

References

1. Cresswell, S.N., McCluskey, T.L., West, M.M.: Acquiring planning domain models using LOCM. The Knowledge Engineering Review 28(02), 195–213 (2013)
2. Ghallab, M., Nau, D., Traverso, P.: Automated planning: theory & practice (2004)
3. Grant, T.: Identifying Domain Invariants from an Object-Relationship Model. PlanSIG2010 p. 57 (2010)
4. Hoffmann, J.: The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables 20, 291–341 (2003)
5. Jilani, R., Crampton, A., Kitchin, D.E., Vallati, M.: ASCoL: Automated acquisition of domain specific static constraints from plan traces. In: The UK Planning and Scheduling Special Interest Group (UK PlanSIG) 2014 (2014)
6. Jilani, R., Crampton, A., Kitchin, D.E., Vallati, M.: Automated Knowledge Engineering Tools in Planning: State-of-the-art and Future Challenges. In: The Knowledge Engineering for Planning and Scheduling workshop (KEPS) (2014)
7. Jilani, R., Crampton, A., Kitchin, D.E., Vallati, M.: ASCoL: a tool for improving automatic planning domain model acquisition. In: Proceedings of the 14th Conference of the Italian Association for Artificial Intelligence (AI*IA 2015) (2015)
8. Plch, T., Chomut, M., Brom, C., Barták, R.: Inspect, edit and debug PDDL documents: Simply and efficiently with PDDL studio. System Demonstrations and Exhibits at ICAPS pp. 15–18 (2012)
9. Shahaf, D., Amir, E.: Learning partially observable action schemas. In: Proceedings of the national conference on artificial intelligence (AAAI) (2006)
10. Vaquero, T.S., Romero, V., Tonidandel, F., Silva, J.R.: itSIMPLE 2.0: An Integrated Tool for Designing Planning Domains. In: Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS). pp. 336–343 (2007)
11. Wickler, G.: Using planning domain features to facilitate knowledge engineering. KEPS 2011 (2011)
12. Yang, Q., Wu, K., Jiang, Y.: Learning action models from plan examples using weighted max-sat. Artificial Intelligence 171(2), 107–143 (2007)
13. Zhuo, H.H.: Crowdsourced Action-Model Acquisition for Planning. In: Proceedings of the AAAI Conference on Artificial Intelligence (2015)