Learning Static Constraints for Domain Modeling from Training Plans

Rabia Jilani
School of Computing and Engineering
University of Huddersfield
United Kingdom

Introduction

AI Planning is a pivotal task that has to be performed by every autonomous system. The automated planning (AP) community has demonstrated a need to uplift planning systems from toy problems to capture more complex domains that closely reflect real life applications (e.g. planning space missions, fire extinction ion management and operation of underwater vehicle) - a way to satisfy the aims of Autonomic systems. Generally, AP techniques require correct description of the planning task. These descriptions include the action model that can be executed in the environment, the state of the objects in the environment and the goal to accomplish.

Domain models encode the knowledge of the domains in terms of actions that can be executed and relevant properties. In centralized approach, this domain could be represented as a knowledge base and automated logical reasoning could be used to determine acts. Specifying operator descriptions by hand for planning domain models is time consuming, error prone and still a challenge for the AI planning community.

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 (Vaquero et al. 2007) and GIPO (Simpson, Kitchin, and McCluskey 2007). A review of the state of the art is provided by Shah et al. (Shah et al. 2013). Recently, also crowdsourcing has been exploited for acquiring planning domain models (Zhuo 2015). 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. Significant differences can be found in terms of the quantity and quality of the required inputs.

Our research concerns the area of automated acquisition of full or partial domain model from one or more examples of action sequences within the domain under study. The aim is to enhance the LOCM system and to extend the method of Learning Domain Models for AI Planning Engines via Plan Traces first published in ICAPS 2009 by Cresswell, McCluskey and West (Cresswell, McCluskey, and West 2013a). This method is unique in that it requires no prior knowledge; however it can produce a complete domain model from training data i.e. plan traces. As compared to LOCM, other systems require more input assistance. ARMS (Yang, Wu, and Jiang 2007), for example, is a system that learns the domain model in addition to domain constraints and invariants. It makes use of background information as input e.g. predicate definitions of initial and goal states. Similarly SLAF (Shahaf and Amir 2006) learns action schema but also requires definitions of fluents and a partial observation of intermediate states as input. For a detailed overview, the interested reader is referred to (Jilani et al. 2014).

The main drawback of LOCM is that it does not produce static knowledge, and its model lacks certain expressive features. A key aspect of research presented in this abstract is to enhance the technique with the capacity to generate static knowledge. A test and focus for this research is to make LOCM able to learn static relationships in fully automatic way in addition to dynamic relationships which LOCM already learn. As per our knowledge, no domain learning system has previously been developed with the aim to learn merely from a set of action traces.

Research contributions include (i) a development of new approach to effectively identify static relations for a wide range of problems, by exploiting graph analysis; using a two staged domain enhancement process that first learn missing static facts for action model and then embed those facts in the partial domain model to get working PDDL domain model (ii) Rule extraction from both optimal and suboptimal plan traces (iii) A useful debugging tool for improving existing models, which can indicate hidden static relations helpful for pruning the search space (iv) Combined with LOCM, ASCoL can be a useful tool to produce benchmark domains (v) Identification of basic categories of Static facts and its impact on heuristic learning systems.

Learning Problem Definition

Domain-independent planning systems require that domain constraints and invariants are specified as part of the input domain model. In AI Planning, the generated plan is correct provided the constraints of the world in which the agent is operating are satisfied. Specifying operator descriptions by hand for planning domain models that also require domain specific constraints is time consuming, error prone and still a challenge for the AI planning community.

LOCM

The LOCM systems perform automated generation of the dynamic aspects of a planning domain model, i.e. changes in the state of the world occurring due to action execution, by considering a set of plan traces, only. A plan trace is a sequence of actions that when applied in an initial state, reach the desired goals. No additional knowledge about initial, goal or intermediate states is needed by LOCM. In comparison with other systems, LOCM approach require a minimal amount of information; other systems also require at least partial state information.

LOCM is based on the assumption that the output domain model can be represented in an object-centred representation (Cresswell, McCluskey, and West 2013b). Using an object-centred representation, LOCM outputs a set of parameterized Finite State Machines (FSMs) where each FSM represents the behaviour of each object in the learnt action schema. Such FSMs are then exploited in order to identify pre- and post-conditions of the domain operators. Although LOCM requires no background information, it usually requires many plan traces for synthesizing meaningful domain models. Moreover, LOCM is not able to automatically identify and encode static predicates.

One drawback of the LOCM process is that it can induce only a partial domain model which represents the dynamic aspects of objects and dose not identify and encode static aspects. Static aspects can be seen as relations that appear in the preconditions of operators only, and not in the effects. Therefore, static facts never change in the world, but are essential for modelling the correct action execution. This is problematic since most domains require static predicates to both restrict the number of possible actions and correctly encode real-world constraints. This is the case in well-known benchmark domains like Driverlog, in which static predicates represent the connections of roads; the level of floors in Miconic, or the fixed stacking relationships between specific cards in Freecell. Any missing static relations are manually introduced into the domain models provided by the LOCM systems. LOCM manually declares this information in the following form:

$$Static(connected(L1, L2), Drive(L1, L2)).$$

The above mentioned Prolog code line is added manually to the input training data file to make it a part of the output domain model manually. The fact in the first argument of static is added as a precondition of the action in the second operator argument of static, where shared variable names provide the binding between the action and its precondition.

ASCoL

We enhance the output domain model of the LOCM system to capture static domain constraints from the same set of input training plans as used by LOCM to learn dynamic aspects of the world. In this research, a new framework ASCoL (Automated Static Constraint Learner) has been presented, to make constraint acquisition more efficient, by observing a set of training plan traces. Most systems that learn constraints automatically do so by analysing the operators of the planning world. The ASCoL system discovers static constraints by analysing plan traces for correlations in the data. To do this the algorithm analyses the complete set of input plan traces, based on a predefined set of constraints, and deduces facts from it. It then augment components of the LOCM generated domain with enriched static constraints.

```
sequence_task(1, [unstack(b8, b9),
stack(b8, b10), pick-up(b7), stack(b7,
b8), unstack(b9, b1), put-down(b9),
unstack(b1, b3), stack(b1, b9),
unstack(b3, b2), stack(b3, b6),
pick-up(b5), stack(b5, b3), unstack(b7,
b8), stack(b7, b2), unstack(b8, b10),
stack(b8, b7), pick-up(b10), stack(b10,
b5)], _, _).
```

Figure 1: An example of a blocksworld plan formatted as required by LOCM.

We define the learning problem that ASCoL addresses as follows. Given the knowledge about parameter types (T), operators' dynamic definition and predicates (fluents) in the form of a PDDL representation of a partial domain model induced by LOCM, and a set of plan traces (P), how can we automatically identify the static relation predicates that are needed by operators' preconditions? We base our approach on the assumption that the input plan traces contain tacit knowledge about constraints validation/acquisition. Based on that assumption, we can draw correlations in the data by pattern discovery in the training input only.

Specifically, the input to the static constraints learning algorithm ASCoL is specified as a tuple (P, T), where P is a set of plan traces (goal directed or random walk) and T is a set of types of action arguments in P which ASCoL parses from LOCM output. ASCoL does not require dynamic knowledge of the domain generated by LOCM. ASCoL accepts input plans (plan traces) in the same text-based format supported by LOCM i.e. a training sequence of N actions in order of occurrence, which all have the form:

$$A_i(O_{i1}, ..., O_{ij}) \quad \text{for} \quad i = 1, ..., N$$

Where A is the action name and O is the actionâs object name. Each action (A) in the plan is stated as a name and a list of arguments. In each action (Ai), there are j arguments where each argument is an object (O) of the problem.

Each plan trace is a sequence of actions (Figure 1) in the order of occurrence to satisfy some goal, where each action in the sequence contains the name of the action and objects that are affected by that action execution. Input plan traces do not include any initial, goal or intermediate states or constraints. Static constraints are to be learnt by the system.

The objective is to learn a complete set of static preconditions, and embed them into the correct operators in the LOCM-learnt output to enrich the domain with this required static knowledge. We formally define the correctness of the learnt static knowledge by comparison with benchmark domains from past IPCs.

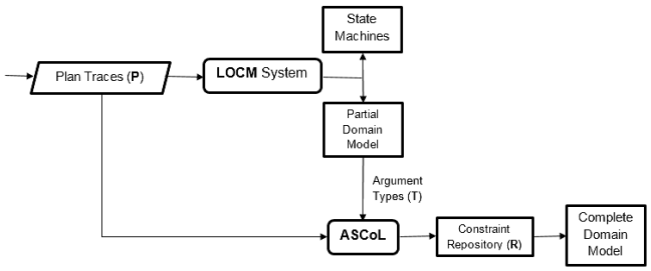

Figure 2: Input Output Structure of ASCoL.

The output for a learning problem is a constraint repository R in PDDL representation that stores all admissible constraints on the arguments of each action A in plan traces P. We assume that input plan traces are noise free while the input domain file at least contains type information for all those operators that the algorithm aims to enhance. Figure 2 shows the general structure of ASCoL in terms of its inputs and outputs.

ASCoL works as a separate unit from LOCM in that LOCM first produces a domain model using a set of plan traces as input. The same LOCM generated domain model, along with the same set of input plan traces, will then be fed to ASCoL to first anticipate the required set of constraints, analyse plan traces and then learn constraints. Finally, it embeds these constraints into the correct operators in the LOCM-learnt output to enrich the domain with this additional static knowledge.

Providing domain constraints to the planning engine may help the planning system in the quick and efficient pruning of the search tree. We aim to capture two major kinds of constraints: domain specific and domain independent constraints. By domain specific we mean the static knowledge that is strictly associated with a domain and is not found as a general example, e.g. the fixed relationships between specific cards in Freecell. Domain independent constraints describe the static knowledge that is generally associated with almost all domains in one way or another; non-equality constraints and link constraints for example.

The ASCoL Methodology

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 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.

Types of Static Facts

We describe different kinds of static facts that we came across whilst experimenting with a variety of domains and modelling strategies. We broadly divide static relations into six different types depending upon the structure of knowledge they contain:

1. Locations Map: facts used for representing an underlying map of connected locations. Relations of this kind are mentioned as `adjacent, link, next, path` or `connected` in the domains including TPP, Zenotravel, Storage, Logistics, Mprime, Spanner, Gripper, Trucks and Gold-miner.

2. Level of Specific Object: parameter objects include varying levels of goods, fuel, space and time depending on the scenario of the domain. Domains that mention such static facts include Mprime, Barman, Trucks, TPP and ZenoTravel.

3. Unordered Sequence: ascending or descending sequence of objects but not in any specific flow. The best example is the sequence of floors in the Miconic domain, where a person can board from any floor and can depart on any other floor (up or down).

4. Ordered Sequence: mentioned as `successor, next` and `link` in different domains including specifically Freecell and other card game domains. Here, such static facts allow the sequential arrangement of cards in card stacks among columns, reserve cells and home cells.

5. Compound Relations: static relations –usually exploited in the encoding of card games– that express two independent static relations in terms of one, i.e `can-stack (card1 card2)`. The intuition behind this is the conventional stacking rule based on card suit and rank order. But this can be decomposed into two separate static facts that can then fulfil our criteria of graph analysis, i.e `can-stack-rank(rcard1 rcard2)` and `can-stack-suit(scard1 scard2)`.

6. Non-equality: the best example for these kinds of static facts are Ferry domain and other general transportation domains where two location objects of travel (obviously of the same type) should be unique.

For a better understanding of the complexity of the learning problem faced by ASCoL, we created metrics to learn the amount of input plans and types of plans (goal and random walk) required to effectively learn these static facts. The same set of planning instances have been used for generating both goal-oriented (GO) and random walk (RW) traces. The later have been created by using a Java-based tool able to parse a given domain model and problem and generate subsequent valid traces of a given length. Clearly, such traces usually do not reach the goal required by the planning instance, but usually provide richer information in terms of the number of transitions for different types of static facts when compared to the goal-oriented plan sequences. Goal-oriented solutions are generally expensive in that a tool or a planner is needed to generate a large enough number of correct plans to be used by the system, but they can also provide useful heuristic information.

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 (Jilani et al. 2015). All domains but Gripper, Logistics and Hanoi, exploit static relations. Input plans of these domains have been generated by using the Metric-FF planner (Hoffmann 2003) 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.

Considering a classification terminology, we can divide the relations identified by ASCoL in to four classes: true positive, true negative, false positive and false negative.

True positive These are correctly identified static relations. Relations identified by ASCoL are almost always static relations which are included in the original domain models.

True negative Dynamic relations that are (correctly) not encoded as static relations. ASCoL did not identify a static relation between arguments that are actually connected by a dynamic relation in any of the considered domains.

False positive It indicates additional relations that actually do not exist in benchmark domains. In some domains ASCoL infers one or two additional relations that are not included in the original domain model. From a Knowledge Engineering point of view, and considering the fact that such additional preconditions do not reduce the solvability of problems, such inferred relations can add value to the original model in terms of effectiveness of plan generation. This is the empirical finding limited to the domains considered for experimentation.

False negative Facts that exist and system dose not identify them. In Freecell and Gold-miner domains ASCoL does not identify all of the static relations.

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

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

Domain	# Ops	# SR	LSR	ASR	CPU-time
TPP	4	7	7	0	171
Zenotravel	5	4	6	2	109
Miconic	4	2	2	0	143
Storage	5	5	5	0	175
Freecell	10	19	13	0	320
Hanoi	1	0	1	1	140
Logistics	6	0	1	1	98
Driverlog	6	2	2	0	35
Mprime	4	7	7	0	190
Spanner	3	1	1	0	144
Gripper	3	0	1	1	10
Ferry	3	1	2	1	130
Barman	12	3	3	0	158
Gold-miner	7	3	1	0	128
Trucks	4	3	3	0	158

Table 1: Overall results on considered domains. For each original domain, the number of operators (# Ops), and the total number of static relations (# SR) are presented. ASCoL results are shown in terms of the number of identified/learnt static relationships (LSR) and number of additional static relations provided (ASR) that were not included in the original domain model. Such relations do not compromise the solvability of problems, but prune the search space. The last column shows the CPU-time in milliseconds for finding static facts for each domain

Table 2 shows, for each considered domain, the percentages of true positive (negative) and false positive (negative) relations identified by ASCoL.

The ability of ASCoL to correctly identify static relations, that should be included as preconditions of specific operators, depends on the number of times the particular operator appears in the provided plan traces. The higher the number of instances of the operator in the plan, the higher the probability that ASCoL will correctly identify all the static relations. We now discuss some of the most interesting results.

Conclusion and Future Goals

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

Domain	TP	TN	FP	FN
TPP	100.0	100.0	0.0	0.0
Zenotravel	100.0	66.6	33.3	0.0
Miconic	100.0	100.0	0.0	0.0
Storage	100.0	100.0	0.0	0.0
Freecell	70.0	100	0.0	30.0
Hanoi	100.0	0.0	100.0	0.0
Logistics	100.0	0.0	100.0	0.0
Driverlog	100.0	100.0	0.0	0.0
Mprime	100.0	100.0	0.0	0.0
Spanner	100.0	100.0	0.0	0.0
Gripper	100.0	0.0	100.0	0.0
Ferry	100.0	50.0	50.0	0.0
Barman	100.0	100.0	0.0	0.0
Gold-miner	33.3	100.0	0.0	66.6
Trucks	100.0	100.0	0.0	0.0

Table 2: For each considered domain, the percentage of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) static relations identified by ASCoL.

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 â is usually missed, since it can hardly be derived by observing transitions only.

In this research we briefly present 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.

We are considering several paths for future work. Grant, in (Grant 2010), 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

Cresswell, S. N.; McCluskey, T. L.; and West, M. M. 2013a. Acquiring planning domain models using LOCM. *The Knowledge Engineering Review* 28(02):195–213.

Cresswell, S. N.; McCluskey, T. L.; and West, M. M. 2013b. Acquiring planning domain models using locm. *The Knowledge Engineering Review* 28(02):195–213.

Grant, T. 2010. Identifying Domain Invariants from an Object-Relationship Model. *PlanSIG2010* 57.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. 20:291–341.

Jilani, R.; Crampton, A.; Kitchin, D. E.; and Vallati, M. 2014. Automated Knowledge Engineering Tools in Planning: State-of-the-art and Future Challenges. In *The Knowledge Engineering for Planning and Scheduling workshop (KEPS)*.

Jilani, R.; Kitchen, D.; Crampton, A.; and Vallati, M. 2015. Learning static constraints for domain modeling from training plans. In *the 14th Conference of the Italian Association for Artificial Intelligence (AI* IA 2015)*, 31.

Shah, M.; Chrpa, L.; Jimoh, F.; Kitchin, D.; McCluskey, T.; Parkinson, S.; and Vallati, M. 2013. Knowledge engineering tools in planning: State-of-the-art and future challenges. In *Proceedings of the Workshop on Knowledge Engineering for Planning and Scheduling*.

Shahaf, D., and Amir, E. 2006. Learning partially observable action schemas. In *Proceedings of the national conference on artificial intelligence (AAAI)*.

Simpson, R. M.; Kitchin, D. E.; and McCluskey, T. 2007. Planning domain definition using gipo. *The Knowledge Engineering Review* 22(02):117–134.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007. itSIMPLE 2.0: An Integrated Tool for Designing Planning Domains. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 336–343.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence* 171(2):107–143.

Zhuo, H. H. 2015. Crowdsourced Action-Model Acquisition for Planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.