Envisioning a Domain Model Learning Track for the IPC

Gregor Behnke,[1] **Pascal Bercher** [2]

[1] Institute for Logic, Language, and Computation, University of Amsterdam
[2] School of Computing, The Australian National University
g.behnke@uva.nl, pascal.bercher@anu.edu.au

Abstract

ICKEPS (International Competition on Knowledge Engineering for Planning and Scheduling) was created to make aware of the importance of domain engineering for planning and scheduling, but in past editions humans competed with the help of planning tools, thus encouraging the development of those tools (where humans were an integral part). We propose an IPC Domain Model Learning track, where learning algorithms would compete completely on their own, creating valid domain models without further use of input other than the constraints (like input plans) they base upon. We believe that this might help to establish some standard in the field of domain model learning, such as a standard benchmark set, standard inputs (possibly an input language), and metrics to evaluate the learned domains against.

Introduction

Planning research is mostly focussed on developing planners, i.e., algorithms that derive plans from a (typically) symbolic planning domain model. However for planning to become attractive to practitioners, they not only require ready access to planners but must also be able to create domain models with ease. Creating them is however a complex knowledge engineering task requiring training and expert knowledge. One means to remedy this problem is to use *Domain Model Learning* – i.e. autonomously inferring a suitable domain model given samples from the domain. In the past, there has been a substantial body of research in domain model learning, including both practical as well as theoretical aspects. Unfortunately, these approaches are rarely comparable on an equal footing as they don't make the same assumptions or use different benchmark sets and metrics to compare them.

We believe that an IPC track on Domain Model Learning might help to establish some cohesion in this area and might lead to an increased interest of practitioners. The proposed track is in spirit similar to ICKEPS (International Competition on Knowledge Engineering for Planning and Scheduling), but sets a different stage: while in ICKEPS humans competed with the help of planning tools, in an IPC Domain Model Learning track, learning algorithms would compete completely on their own – setting the same focus as all other IPC tracks.

Why Domain Model Learning?

The usability of planning hinges on the availability of suitable domain models. Creating such models often requires both expertise in planning as well as in the application domain at hand. In successful applications, this has typically happened either by a collaboration between planning researchers and practitioners (Helmert and Lasinger 2010; Vaquero et al. 2012; Kuter et al. 2018) or in highly research-heavy settings (Bresina and Morris 2006; Chien et al. 2015). For planning to be a true success story, the ability to use it – and thus the ability to create suitable domain models – must be made available to a broader audience. This includes, for example, Small and Medium Enterprises (SMEs) that typically cannot afford the financial commitment involved in a research-intensive process of domain modelling.

The approach of domain model learning is one means by which this problem could be solved. Overviews of existing methods are made available by Arora et al. (2018) and online at https://macq.planning.domains by Callanan et al. (2022). The task of domain model learning is to *automatically* infer a planning domain model from given information. Typically, this takes the form of given plans (i.e. action sequences that are supposed to be solutions (Cresswell, McCluskey, and West 2013; Cresswell and Gregory 2011)), full plan traces (i.e. sequences of states traversed by solution plans (Jimenez et al. 2012; Stern and Juba 2017; Aineto, Jiménez, and Onaindia 2018; Arora et al. 2018; Juba, Le, and Stern 2021; Bachor and Behnke 2024)), or partial plan traces with only state features being observable (Yang, Wu, and Jiang 2007; Aineto, Celorrio, and Onaindia 2019).

Some approaches assume an initial (flawed or incomplete) model on top of plans, often referred to as model repair [McCluskey, Richardson, and Simpson, 2002; Lin and Bercher, 2021; 2023; Lin, Höller, and Bercher, 2024]. Some approaches to model repair also do not require input plans or state traces, but instead assume an unsolvable problem and attempt at making it solvable (Göbelbecker et al. 2010; Xiao et al. 2020; Gragera et al. 2023) – we are however not aiming at testing such works in this first edition. Also worth noting in this context (despite us not aiming at testing this in the proposed competition) is the work on model reconciliations (Sreedharan, Chakraborti, and Kambhampati 2021; Sreedharan, Bercher, and Kambhampati 2022), which, from an abstract point of view, can also be referred to as model repair, but bases on two given domain models and aims at making changes to one of them, so that they explain the optimality of a given plan that happened to be optimal in the other model already.

Most learners however learn the entire model from scratch based on given plans/plan traces. Such plans or plan traces can be expected to be easily generated or provided by application experts, e.g., by describing the current workflow of a factory or current lab procedures. Lin, Grastien, and Bercher (2023) argue that this approach of providing plans (on top of the existing model) is similar to the test and refine paradigm known from programming and hence a canonical approach to domain verification and repair as well.

Why a Competition on Domain Model Learning?

While there has been some interest into domain model learning in the past, and some successful methods have been developed, research in domain model learning is quite diverse and not easily usable in practice.

Typically, a competition establishes common ground for the community in terms of (1) what to research, (2) which problems to tackle, and most importantly (3) how to compare different methods. More concretely, a competition first of all requires a common understanding and definition of what the problem of "domain model learning" to be tackled actually is. This definition also helps to highlight nuances and variants in domain model learning that have so far been overlooked. Similarly, it makes the capabilities of the individual domain model learners clear to other researchers and practitioners and thus enables a more fair comparison. Next, a competition would also yield a benchmark set for domain model learning that can be used in the future to compare newly developed domain model learners. This also includes the formal definition of a common input language. Lastly, the competition will also propose and use metrics to compare the learners, which are then likely to be picked up by other researchers.

What are the Inputs for Domain Model Learning?

Typically, domain model learners take plans as their input. While some learners focus on learning from a single plan, typically, multiple plans are assumed as their input. However other modes or models are also conceivable. Notably, one might consider to also provide negative examples of plans, i.e., sequences of actions that should not be plans in the learned domain. Providing such negative examples is connected to the notion of justifications that we discuss on page 4. For example, when providing a plan π to learn from, we might provide the guarantee that this plan is optimal for the domain to learn. This implicitly induces an exponential amount of non-plan examples as well – as no shorter action sequence can be a plan. Similarly, it could also be conceivable to learn only from action sequences of length one, i.e., from triples of a state, an action, and a successor state – as this is a format that a practitioner could produce reasonably quickly.

How these plans look like is a matter of system capabilities: The more information there is in the input, the more can be used and the less the system must hypothesise about. In practice, we believe a wide spectrum of possible inputs makes sense here – depending on what the modeler already knows. Some possibilities include:

- Action names without any parameters (the least information possible)
- Action names with full groundings, i.e., parameter instantiations (the most information possible when disregarding preconditions/effects)
- Action names with partial parameter instantiations or restrictions on parameters (partial knowledge is available, e.g., that a vehicle type can be restricted to a car (type), but it's still not clear which concrete car)
- No action names at all, but we are given the full graph of the transition system (Bonet and Geffner 2020).
- Any of the above, but with a pre-existing model, i.e., actions do already have preconditions and effects that might have to be extended or exchanged (making this a model repair problem).

In theory, even generalizations of the above might be conceivable. For example, rather than providing an exact plan, an agent might provide some sort of generalization thereof, of which such a plan might just be a special case. For example, instead of providing a specific action at a position, a disjunction of actions could be provided, thus modeling insecurity about what exactly has been observed. This however only seems to make sense where plans are either "observed" (with uncertainty) rather than generated by the modeler him/herself. Similar situations have already been discussed in plan and goal recognition (Keren, Gal, and Karpas 2020; Meneguzzi and Pereira 2021) – when e.g. partial observability of the performed actions is discussed. For the sake of simplicity, a first competition on domain model learning should however not consider too many settings here.

Most approaches then additionally assume that the states that result from the execution of actions are at least partially given (Yang, Wu, and Jiang 2007; Jimenez et al. 2012; Stern and Juba 2017; Arora et al. 2018; Aineto, Jiménez, and Onaindia 2018; Aineto, Celorrio, and Onaindia 2019; Juba, Le, and Stern 2021).

On the other hand LOCM (Learning Object-Centred Models) and LOCM2 (Gregory and Lindsay 2016) assume that no state information is given and thus have to infer the existence of state features. Similarly, the work by Bonet and Geffner (2020) assumes that the input contains the set of states, but these states do not have any internal factored structure, which needs to be inferred.

As mentioned earlier, several systems (Lin, Grastien, and Bercher 2023; Lin, Höller, and Bercher 2024) and theoretical investigations (Lin and Bercher 2021, 2023) additionally assume an already partial model to be provided (stemming either from a domain modeler or novel tools, as based on LLMs (Oswald et al. 2024a)), which has to be repaired based on the additional input (i.e., plans). While these approaches can learn from an "empty domain", the inverse is not true for the other approaches – they typically can't utilise a given domain. It is worth pointing out that yet no approach is a special case of the other: current approaches to repair an existing domain cannot learn new state features (which some classical learning approaches can), yet these classical learning approaches can't take existing models into account making both approaches distinct lines of research (though with the potential of being combined).

Given that creating initial versions of planning domains has become much easier due to the usage of Large Language Models (LLMs) or Natural Language Processing-based techniques in general (Lindsay et al. 2017; Hayton et al. 2020; Guan et al. 2023; Oswald et al. 2024a), we believe that the assumption of having such an initial model is very reasonable – while at the same time it also shows the necessity for validating and improving them based on further explicit constraints provided by the user (such as plans or plan places).

Further, it is also conceivable that the plans given as input to the learners are in some sense noisy. While most learners assume that any given information inside of the plans is correct, some can capture unreliable inputs (Mourão et al. 2012; Zhuo and Kambhampati 2013; Lamanna and Serafini 2024). This uncertainty can either be in the observed state being unreliable (Mourão et al. 2012; Lamanna and Serafini 2024), or in the plan itself (Zhuo and Kambhampati 2013), where either actions could be missing, the wrong actions, or wrong parameters could have been observed.

Lastly, most learners assume that all plans from which the model should be learned are given in one batch. It is however also reasonable to first give the planner a batch of plans and then allow the planner to query for additional information. This could, e.g., be a specific plan where the planner inquires for whether this is an intended solution to the planning problem. This way the learner is allowed to acquire the necessary information to refine the domain model itself. These approaches are called online learning approaches (Lamanna et al. 2021), while the others are called offline learning.

We would like to note that the allowed input plans put some restrictions on the type of the model to be learned. E.g., if STRIPS or PDDL-style state descriptions are given, a corresponding model has to be learned. This for examples excludes (without additional effort) to learn a Factored Transition System model (Torralba and Sievers 2019) or an SAS+ model (Bäckström and Nebel 1995). Similarly, if the plan contains numeric elements (e.g., natural numbers as arguments for actions), the learners are forced to learn numeric models, which is potentially more difficult.

On which of these approaches (learning the entire model based on action names only vs. repairing existing models) to focus on – or potentially both – in a competition would be subject to interest in the community.

What are the Outputs for Domain Model Learning?

While at first glance it might look clear or obvious what the desired output should be, i.e., what exactly it is that should be learned, we believe that this deserves further discussion – but at least it should be clearly specified. The first major distinction is whether a grounded or lifted model should be learned. For the grounded case, the learner can specify separate preconditions and effects for every action occurring in the given plans. For the lifted case, it has to provide a PDDL-style domain model that fits all ground actions given in the plans. Here the amount of information given in the input plays a role: if no, or not all action parameters are given, the learner also has to decide on the actual parameter values for these hidden parameters and thus on the action ground instances occurring in he plan.

The most obvious "output" to learn is:

- all preconditions and effects of the input plans provided,
- additional actions (Bonet and Geffner 2020), which makes sense when only the transition system of the problem is given but no action labels,
- in particular when a (partial) model to repair is given, the question arises what may be *changed*: can only preconditions and effects be *added*, or also exchanged/removed? Can specific parts of the model be "blocked" from being changed (since, e.g., the modeler expresses confidence that this particular part is definitely correct or important). Can action parameters be added/changed/deleted? For example, can action parameter types be changed (e.g., more restricted)? The latter would also be up to debate in the other settings where no preconditions/effects are provided but only actions names and their parameters.

Similar to the question of what the allowed input should be, the question about the computed output should likely be discussed with the community to make sure that the very first competition doesn't exclude anybody interested. So we'd likely aim for the least common denominator – but want to engage with the community to see what is desired the most.

How to Evaluate Domain Model Learning?

Unfortunately, there is no standard benchmark set on which domain model learning approaches are tested. The usual way that researchers compare their approaches is the following:

1. Select a set of domains, normally from the IPC, and declare them as the "ground truth"
2. Generate plans for the planning problems of this domain (either for randomly generated problems or for the official IPC problems)
3. Learn a domain from these plans
4. Compare the learned domain syntactically to the original ground truth domain

This syntactic comparison mostly takes the form of computing the intersection and differences between the two models. I.e. for every action a, one would compute $|pre_{ground}(a) \cap pre_{learned}(a)|$ as well as $|pre_{ground}(a) \setminus pre_{learned}(a)|$ and $|pre_{learned}(a) \setminus pre_{ground}(a)|$ (same for effects), and use this as a score.

This metric has two severe problems. Firstly, the metric compares syntax and not semantics. It is often possible to describe the same domain with two syntactically different descriptions. This is already evidenced by the large body on research on domain re-writing, reformulation, and transformations . Thus, a syntactic metric at the end requires the domain model learner to obtain the same syntactic representation of the problem as the original ground truth domain did. It is possible to relax this requirement slightly by using the graph edit distance on a graph describing the lifted model (Chrpa et al. 2023), which is able to take some syntactic isomorphisms between predicates and objects into account. It is however limited to cases where actions are still functionally bijective between the two models. However the learner has no information available to make any sensible inference w.r.t. the unknown ground truth model.

Secondly, at its core, this metric does not evaluate the domain model learner, but the set of sample plans that have been drawn. Even a hypothetical "perfect" domain model learner could get a bad score – if it is simply impossible to derive the ground truth. Likewise, somebody who "inferred" the ground truth would get a perfect score, despite potentially just have been lucky.

Comparing two learners on this basis amounts to nothing but evaluating either a random guess or an implicit inductive bias of the learner – which in both cases, does not reveal any information about the learner.

For the first problem, recent papers have proposed to compare domains in a semantic way instead of a syntactic way. One option is to only consider the reachability between possible initial states and goals. For this we consider the set of all pairs of initial and goal states (s_I, s_G) so that there is a plan leading from s_I to s_G. Two lifted domains are *functionally equivalent* if for all sets of objects, the set of these initial-goal pairs is the same (Shrinah, Long, and Eder 2021). To only consider sensible initial states (e.g., not ones where a transporter is at two locations at the same time), one might consider to restrict this definition to initial states only that satisfy a description of possible initial states (Grundke, Röger, and Helmert 2024). When it comes to domain learning, this definition could be somewhat problematic as it allows for any transformation (renaming, splitting, merging) of actions inside the models – not even the number of actions in both models has to be the same. As such, while we think this approach is reasonable, we deem it too lax for a competition as it allows learners do deviate too much from given traces in terms of how which actions the model contains.

Another approach is to compare domains by the plans that solve their planning problems – essentially a more concrete version of Shrinah, Long, and Eder's functional equivalence (Oswald et al. 2024b; Vallati and Chrpa 2019). Ideally, one would consider two domains are equivalent, if their set of plans that solve them is the same, i.e., them being *plan equivalent* (Vallati and Chrpa 2019). While computing the set of plans is impossible, since this is potentially infinite, one can regard each problem as a formal language (Höller et al. 2014; Höller et al. 2016) and check whether those are identical – by checking whether their "automata" or "grammars" are identical, i.e., the underlying transition system. This is what Chrpa et al. (2023) do: in case these are not identical, they compute the edit distance (based on Answer Set Programming).

While an edit distance on the domain might be a theoretically interesting approach, we would like to highlight a practical consideration: For someone applying domain model learning it is important that (1) the learner does not generate a domain that admits invalid plans (what Juba, Le, and Stern (2021) call *safe domain model learning*) and that (2) the learned domain does not exclude actually valid plans. The notion of an edit distance does not correlate with this intuitive notion: even a minimal change to a domain (think: removing the deleting effect on a `move` action) can have a catastrophic effect of the practical usefulness of the learned domain. In essence: not all edits are made equal – some edits to a domain are more problematic than others. An alternative approach could use an approximation of the set of plans that a domain admits: We only consider the k shortest non-redundant plans of the domain – determining these plans is typically called top-k planning (Katz et al. 2018; Speck, Mattmüller, and Nebel 2020) and planners able to find non-redundant top-k plans are available (von Tschammer, Mattmüller, and Speck 2022). Using a top-k planner, these sets can be efficiently determined for both the "ground truth" and the learned domain. We can then use the size of the intersection and set differences between these sets as a metric for the quality of the learned domain. One author has already used this method successfully in teaching.

The second problem is however much harder to resolve. The optimal solution here would be to formally define which conclusions about the original domain can be drawn based on a set of sample plans and then only evaluate this inference. (This is in line with what Lin and Bercher (2021) do provided an initial model is given, as they compute the minimal number changes required to the model that are in line with the sample plans. Though this isn't a satisfying solution either as it only guarantees to find a model with minimal edit distance, but it might still not be the perfectly desired one.)

This is however only possible in very controlled settings. One such idea was recently presented by Bachor and Behnke (2024). Here, the given plans come with a guarantee of non-redundancy, namely well or perfect justification (Fink and Yang 1992). A plan is well justified, if removing any one action makes it invalid (either non-executable or non-goal-achieving), while it is perfectly justified if removing any subset of its actions makes it invalid. To evaluate a learner in this setting, we would still generate a set of plans Π from a "ground truth", s.t. they are either well or perfectly justified. We would then forget the original domain and only use Π as a metric. The domain model learner's task is then to infer a domain $\mathcal{D}$. To test the validity of this domain, we then determine for every plan $\pi \in \Pi$ whether it is well or perfectly justified in the learned domain $\mathcal{D}$. If it is, the learned domain is correct, and incorrect otherwise.

For a competition, this setting might however not be practical, as the learners have to be specifically designed for this setting. Instead we could use a stochastic evaluation. If we choose the set of sample plans large enough, it is likely that these plans will unique characterise the ground truth domain. We could thus let the learner learn a domain from multiple sets of samples for the same ground domain. On average, learners with higher "correctness" are then expected to learn the correct domain model more often. I.e. we could use the success rate or the mean accuracy (based on the above metrics).

Another route would be to let the learners allow to decide themselves how many samples they need. We again assume a ground truth domain Π. The learners can then draw sample plans from this domain Π until they are satisfied that they have drawn enough samples to be able to infer a learned domain Π^{ℓ} that is semantically equivalent to Π (i.e. both have the same set of plans). The score of a planner in this setting would be the number of samples it needs to characterise the correct domain.

In case a domain model is already provided, another question is how this is taken into account. Lin, Grastien, and Bercher (2023) and Lin, Höller, and Bercher (2024) chose to find the minimal number of repairs that make the resulting model compatible with the requested properties (such as the demanded plans), though there are several other metrics that could make sense, for example not reducing the set of plans that the original model admitted, not increasing it (beyond the ones provided), and many more. So there is no single evaluation metric that's "the right choice", there are several one may argue for, and possibly several of them could be offered, e.g., in different tracks similar to the standard IPC where coverage and agile (often referred to as the IPC score, i.e., $\min\{1, 1 - \frac{\log(t)}{\log(t)}\}$ for time limit T and runtime t) are two of the best known ones.

Conclusion and Next Steps

In this paper we proposed to establish a track on learning domain models at the IPC. This might address the issue of missing cohesion in this line of research, of the various approaches differ in assumptions, use different benchmark sets, and metrics to compare them. That said, several of these questions still had to be clarified for such a competition, most notably what the input is (e.g., plans, non-plans, plan traces, a partial (flawed) initial model to repair) and how to evaluate the learned domain. A possible way to resolve these open questions is by branching some of the most canonical and useful answers to these questions into multiple tracks.

Acknowledgements

Pascal Bercher is the recipient of an Australian Research Council (ARC) Discovery Early Career Researcher Award (DECRA), project number DE240101245, funded by the Australian Government.

References

Aineto, D.; Celorrio, S. J.; and Onaindia, E. 2019. Learning action models with minimal observability. *Artificial Intelligence*, 275: 104–137.

Aineto, D.; Jiménez, S.; and Onaindia, E. 2018. Learning STRIPS Action Models with Classical Planning. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS 2018)*, 399–407. AAAI Press.

Arora, A.; Fiorino, H.; Pellier, D.; Metivier, M.; and Pesty, S. 2018. A Review of Learning Planning Action Models. *The Knowledge Engineering Review*, 33: 1–25.

Bachor, P.; and Behnke, G. 2024. Learning Planning Domains from Non-Redundant Fully-Observed Traces: Theoretical Foundations and Complexity Analysis. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI 2024)*, 20028–20035. AAAI Press.

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS+ Planning. *Computational Intelligence*, 11(4): 625–656.

Bonet, B.; and Geffner, H. 2020. Learning First-Order Symbolic Representations for Planning from the Structure of the State Space. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*, 2322–2329. IOS Press.

Bresina, J.; and Morris, P. 2006. Mission Operations Planning: Beyond MAPGEN. In *Proceedings of the 2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT 2006)*, 151–156. IEEE.

Callanan, E.; De Venezia, R.; Armstrong, V.; Paredes, A.; Chakraborti, T.; and Muise, C. 2022. MACQ: a Holistic View of Model Acquisition Techniques. In *Proceedings of the 2022 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS 2022)*.

Chien, S.; Rabideau, G.; Tran, D.; Troesch, M.; Doubleday, J.; Nespoli, F.; Ayucar, M. P.; Sitja, M. C.; Vallat, C.; Geiger, B.; Altobelli, N.; Fernandez, M.; Vallejo, F.; Andres, R.; and Kueppers, M. 2015. Activity-based Scheduling of Science Campaigns for the Rosetta Orbiter. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 4416–4422. AAAI Press.

Chrpa, L.; Dodaro, C.; Maratea, M.; Mochi, M.; and Vallati, M. 2023. Comparing Planning Domain Models using Answer Set Programming. In *Proceedings of the 18th European Conference on Logics in Artificial Intelligence (JELIA 2023)*, 227–242. Springer.

Cresswell, S.; and Gregory, P. 2011. Generalised Domain Model Acquisition from Action Traces. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS 2011)*, 42–49. AAAI Press.

Cresswell, S. N.; McCluskey, T. L.; and West, M. M. 2013. Acquiring planning domain models using LOCM. *The Knowledge Engineering Review*, 28(2): 195–213.

Fink, E.; and Yang, Q. 1992. Formalizing Plan Justifications. In *Proceedings of the 9th Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI 1992)*, 9–14.

Göbelbecker, M.; Keller, T.; Eyerich, P.; Brenner, M.; and Nebel, B. 2010. Coming up With Good Excuses: What to do When no Plan Can be Found. In *Proceedings of the 20nd International Conference on Automated Planning and Scheduling (ICAPS 2010)*, 81–88. AAAI Press.

Gragera, A.; Fuentetaja, R.; Olaya, Á. G.; and Fernández, F. 2023. A Planning Approach to Repair Domains with Incomplete Action Effects. In *Proceedings of the 33rd Interna-*

tional Conference on Automated Planning and Scheduling (ICAPS 2023), 153–161. AAAI.

Gregory, P.; and Lindsay, A. 2016. Domain Model Acquisition in Domains with Action Costs. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 149–157. AAAI Press.

Grundke, C.; Röger, G.; and Helmert, M. 2024. Formal Representations of Classical Planning Domains. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS 2024)*, 239–248. AAAI Press.

Guan, L.; Valmeekam, K.; Sreedharan, S.; and Kambhampati, S. 2023. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems (NeurIPS 2023)*, 36: 79081–79094.

Hayton, T.; Porteous, J.; Ferreira, J.; and Lindsay, A. 2020. Narrative planning model acquisition from text summaries and descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2020)*, 1709–1716. AAAI Press.

Helmert, M.; and Lasinger, H. 2010. The Scanalyzer Domain: Greenhouse Logistics as a Planning Problem. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS 2010)*, 234–237. AAAI Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2014. Language Classification of Hierarchical Planning Problems. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, 447–452. IOS Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2016. Assessing the Expressivity of Planning Formalisms through the Comparison to Formal Languages. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling, (ICAPS 2016)*, 158–165. AAAI Press.

Jimenez, S.; De La Rosa, T.; Fernandez, S.; Fernandez, F.; and Borrajo, D. 2012. A Review of Machine Learning for Automated Planning. *The Knowledge Engineering Review*, 27(4): 433–467.

Juba, B.; Le, H. S.; and Stern, R. 2021. Safe Learning of Lifted Action Models. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning (KR 2021)*, 379–389. IJCAI.

Katz, M.; Sohrabi, S.; Udrea, O.; and Winterer, D. 2018. A novel iterative approach to top-k planning. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS 2018)*, 132–140. AAAI Press.

Keren, S.; Gal, A.; and Karpas, E. 2020. Goal Recognition Design – Survey. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI 2020)*, 4847–4853. IJCAI.

Kuter, U.; Goldman, R. P.; Bryce, D.; Beal, J.; Dehaven, M.; Geib, C. S.; Plotnick, A. F.; Nguyen, T.; and Roehner, N. 2018. XPLAN: Experiment Planning for Synthetic Biology. In *Proceedings of the First ICAPS Workshop on Hierarchical Planning*, 48–52.

Lamanna, L.; Saetti, A.; Serafini, L.; Gerevini, A.; and Traverso, P. 2021. Online Learning of Action Models for PDDL Planning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*, 4112–4118. IJCAI.

Lamanna, L.; and Serafini, L. 2024. Action Model Learning from Noisy Traces: a Probabilistic Approach. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS 2024)*, 342–350. AAAI Press.

Lin, S.; and Bercher, P. 2021. Change the World – How Hard Can that Be? On the Computational Complexity of Fixing Planning Models. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*, 4152–4159. IJCAI.

Lin, S.; and Bercher, P. 2023. Was Fixing this Really That Hard? On the Complexity of Correcting HTN Domains. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023)*, 12032–12040. AAAI Press.

Lin, S.; Grastien, A.; and Bercher, P. 2023. Towards Automated Modeling Assistance: An Efficient Approach for Repairing Flawed Planning Domains. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023)*, 12022–12031. AAAI Press.

Lin, S.; Höller, D.; and Bercher, P. 2024. Modeling Assistance for Hierarchical Planning: An Approach for Correcting Hierarchical Domains with Missing Actions. In *Proceedings of the 17th International Symposium on Combinatorial Search (SoCS 2024)*, 55–63. AAAI Press.

Lindsay, A.; Read, J.; Ferreira, J.; Hayton, T.; Porteous, J.; and Gregory, P. 2017. Framer: Planning models from natural language action descriptions. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2017)*, 434–442. AAAI Press.

McCluskey, T. L.; Richardson, N. E.; and Simpson, R. M. 2002. An Interactive Method for Inducing Operator Descriptions. In *Proceedings of the 6th International Conference On Artificial Intelligence Planning And Scheduling (AIPS 2002)*, 121–130. AAAI Press.

Meneguzzi, F.; and Pereira, R. F. 2021. A Survey on Goal Recognition as Planning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*, 4524–4532. IJCAI.

Mourão, K.; Zettlemoyer, L.; Petrick, R. P. A.; and Steedman, M. 2012. Learning STRIPS operators from noisy and incomplete observations. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI 2012)*, 614–623. AUAI Press.

Oswald, J.; Srinivas, K.; Kokel, H.; Lee, J.; Katz, M.; and Sohrabi, S. 2024a. Large Language Models as Planning Domain Generators. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS 2024)*, 423–431. AAAI Press.

Oswald, J.; Srinivas, K.; Kokel, H.; Lee, J.; Katz, M.; and Sohrabi, S. 2024b. Large Language Models as Planning Domain Generators (Student Abstract). In *Proceedings of* *the 38th AAAI Conference on Artificial Intelligence (AAAI 2024)*, 23604–23605. AAAI Press.

Shrinah, A.; Long, D.; and Eder, K. 2021. D-VAL: an automatic functional equivalence validation tool for planning domain models. *arXiv preprint arXiv:2104.14602*.

Speck, D.; Mattmüller, R.; and Nebel, B. 2020. Symbolic Top-k Planning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*, 9967–9974. AAAI Press.

Sreedharan, S.; Bercher, P.; and Kambhampati, S. 2022. On the Computational Complexity of Model Reconciliations. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI 2022)*, 4657–4664. IJCAI.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2021. Foundations of Explanations as Model Reconciliation. *Artificial Intelligence*, 301: 103558.

Stern, R.; and Juba, B. 2017. Efficient, Safe, and Probably Approximately Complete Learning of Action Models. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI 2017)*, 4405–4411. IJCAI.

Torralba, A.; and Sievers, S. 2019. Merge-and-Shrink Task Reformulation for Classical Planning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, 5644–5652. IJCAI.

Vallati, M.; and Chrpa, L. 2019. On the robustness of domain-independent planning engines: the impact of poorly-engineered knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture (K-CAP 2019)*, 197–204. Association for Computing Machiner.

Vaquero, T. S.; Costa, G.; Tonidandel, F.; Igreja, H.; Silva, J. R.; and Beck, J. C. 2012. Planning and Scheduling Ship Operations on Petroleum Ports and Platforms. In *Proceedings of the 2012 Scheduling and Planning Applications woRKshop (SPARK 2012)*, 8–16.

von Tschammer, J.; Mattmüller, R.; and Speck, D. 2022. Loopless top-k planning. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS 2022)*, 380–384. AAAI Press.

Xiao, Z.; Wan, H.; Zhuo, H. H.; Herzig, A.; Perrussel, L.; and Chen, P. 2020. Refining HTN Methods via Task Insertion with Preferences. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020)*, 10009–10016. AAAI Press.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence*, 171(2-3): 107–143.

Zhuo, H. H.; and Kambhampati, S. 2013. Action-model acquisition from noisy plan traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2444–2450. AAAI Press.