Author Response Information for Submission 92

Response Letter(s)

Response Letter	
Response:	We thank the reviewers for their insightful comments that will definitely help to improve the contents of the paper. Reviewer#1 We agree with reviewer#1 that scalability is currently an issue. With this regard, we believe that the classical planning compilation is an appealing approach because it enables the use of classical planning heuristics for computing scalable estimates of the P(O|M)P(M) probabilities (this approach is already successful in "Plan Recognition as Planning" for the scalable computation of the P(O|G)P(G) probabilities). We are happy the reviewer agrees with us that this is a natural course of action as evidenced by his second question on the use of relaxed problems. Note that scalability is impacted by the complexity of computing the distance from a model to an observation. This complexity depends on the observation model used, which in our case allows for observations with neither actions nor intermediate states. This flexibility on the observations also motivates the use of planning versus other approaches (more details in answer 2 to reviewer#2). Reviewer#2 Much appreciated the enjoyment of the reviewer with the paper reading. 1) Model Recognition versus Model Learning? While Model Learning induces a model from observations of plan executions, Model Recognition computes the probability of a given model to produce an observation of the plan execution. Hence, Model Recognition can be regarded as a "gray-scale" plan validation that a) is capable of dealing with incomplete plans (partially observed plans) and b) whose output is no longer binary (valid plan/invalid plan) but a probability that the plan is produced by a model.

2) Distances and Model Editing

The P(O|M) likelihood introduced in this paper is defined in terms of the distance from a Model to an Observation. The computation of the distance could be addressed using the empty model variation of Model Reconciliation the reviewer suggests but under the assumption that the observation includes the complete plan. Similarly, Model Updates approaches could also be used under their own limitations. The cited work on observation edits [Sohrabi et al. IJCAI 2016] is of special interest here, since it opens up the way to an alternative distance where instead of editing the model to fit the observation, it is the observation that is edited to fit the model.

Using planning, however, poses some advantages over other model editing approaches. The main advantage being that it is suitable for partially observable plan executions (including observations with no observed actions) since the planner is able to fill the gaps of this incomplete observation. We use an existing model learning approach (AJO in the paper) extended to support our observation model presented in page 2. As indicated in the paper, when an initial model is given, rather than starting from scratch, this learning approach behaves as a model editing system and can be used to compute the distance. The single restriction we impose to the observations is that the initial state and part of the final state are known so, to enumerate a few examples, we can accept as input: observations with no actions, initial/final states pairs, pre-state/post-state pairs, incomplete plans, etc. Note that the length of the plan cannot be inferred from the observation, since full planning steps (<a_i, s_i> pairs) might be missing from the observation. This flexibility could not be achieved using other approaches such as Model Reconciliation.

Furthermore, the compilation to classical planning enables the use of classical planning heuristics. This means that, without additional work, the distance from a model to an observation can be computed: (1) exactly using optimal planners, (2) approximately using satisficing planners, and (3) approximately (very fast) using planning heuristics.

3) Why probabilities?

Model Recognition defines a P(O|M)P(M) Bayesian probability distribution that allows reasoning over different prior probabilities of input models. This kind of 'a priori' knowledge cannot be exploited using simply distances. In fact, using a system purely based on distances (model edits) is equivalent to assuming that the input models are equiprobable. Therefore, probabilities allow for a more general presentation of the Model Recognition framework.

This is also the justification for using a Bernoulli process over a simpler model such as least edits. The Bernoulli process offers the same behavior as a least edits model when the P(M) probabilities are equiprobable, and supports different prior distributions. Moreover, this is with no additional cost since the computation is done in constant time.

	4) “Full observability of the executed plan” means having the plan and all the traversed intermediate states (as indicated in the observation model in page 2). In the grid-world example mentioned by the reviewer, a model with no delete effects produces different intermediate states than a model with delete effects. 5) “Partial observability of the executed plan” Assumption 2 is taken literally from [Ramirez 2012] and also appears in Ramírez&Geffner: Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. AAAI 2010, which we forgot to cite. This assumption means that all the optimal plans that are consistent with the input observation are equivalent. It is only the cost of the optimal plan that matters (personal communication) Reviewer#3 The theoretical model (and the P(O|M) likelihood) are used to define the Bayesian classifier argmax P(O|M)P(M), which outputs the model that most likely produced an observation. Accuracy, on the other hand, is a metric used to evaluate the performance of a classifier over a number of input samples and can be described as "the percentage of samples correctly classified". High values of accuracy are indicative that the assumptions underlying our theoretical model and which allow us to compute the P(O|M) likelihood are indeed adequate abstractions of reality. The ‘Recognizing failures’ part of the experimental section is just an example of a possible application of Model Recognition. In this experiment, each model encodes a different type of failure, so by applying the Model Recognition task one can identify the failure that happened during, for example, a manufacturing process. The source code of all the models used in the experiments is available at folder model-recognition/benchmarks/icaps19/ of the provided anonymous repository.
Time:	Jan 17, 17:08 GMT
Letter:	Dear [*NAME*], We apologize for the duplicate message, but it was necessary to get Easychair to open the review window. Thank you for your submission to ICAPS 2019. The ICAPS 2019 review response period starts now and ends January 17. During this time, you will have access to the current state of your reviews and have the opportunity to submit a response. Please keep in mind the following during this process: * Almost all papers have three reviews. Some may have four. A

	very low number of papers are missing one review. We hope to get that review completed in the next day. We apologize for this. * The deadline for entering a response is January 17th (at 11:59pm UTC-12 i.e. anywhere in the world). * Responses must be submitted through EasyChair. * Responses are limited to 1000 words in total. You can only enter one response, not one per review. * You will not be able to change your response after it is submitted. * The response should focus on what you believe to be errors or misunderstandings in the reviews and responses to any questions posed by the reviewers. Try to be as concise and to the point as possible. * The review response period is an opportunity to react to the reviews, but not a requirement to do so. Thus, if you feel the reviews are accurate and the reviewers have not asked any questions, then you do not have to respond. * The reviewers may not agree on your paper. In many cases, there has already been discussion among the reviewers and some revision of reviews. There will likely be additional discussion and revision after the feedback period. We may also find it necessary to solicit additional reviews in cases where the reviewers disagree. * The program committee will read your responses carefully and take this information into account during the discussions. On the other hand, the program committee may not directly respond to your responses in the final versions of the reviews. The reviews on your paper are attached to this letter. To submit your response you should log on to the EasyChair Web page for ICAPS 2019 and select your submission on the menu. Nir, Eva and David Program Chairs [*REVIEWS*]
Time:	Jan 15, 14:46 GMT

Reviews

Review 1

Significance:	**3**: (substantial contribution or strong impact)
Soundness:	**3**: (correct)
Scholarship:	**2**: (relevant literature cited but could be expanded) Some works have been done in this field by E. Amir and coauthors. In partucilar in learning action models with planning. The work is relevant to this approach, and should be mentioned/compared. E.g. see ``Learning Partially Observable Deterministic Action Models (JAIR 2008)
Clarity:	**2**: (mostly readable with some room for improvement) The paper is clear on average, but certain parts could be better written and made clearer. E.g. The last paragraph in p. 1 (To better illustrate...)
Reproducibility:	**5**: (code and domains (whichever apply) are already publicly available) The code is available on github (but I haven't tested it yet)
Overall evaluation:	**2**: (accept)
Review:	The paper describes a novel approach to The authors propose to state the problem of model recognition (i.e. identifying the action schema of a planning problem from the observations trace) as a planning problem. The approach is interesting, and brings to the next level the work developed in the last years on the topic of planning recognition. What is interesting, is the formalisation of the model, and the planning problem that is built to solve the problem, the Model Recognition as planning. A couple of questions nevertheless arise (that authors might want to answer). 1. There is a scalability issue that is strongly linked to the approach, due to several issues, but mainly, on the size of the generated planning problem. This subject has been approached in many ways in the past, in particular by successive works by E. Amir & coauthors (see Learning partially observable action models: Efficient algorithms). How do the authors foresee future works in that direction? Would if be useful to discard the actions that will not be

	used from the model (because of no observations) maybe by proceeding in an incremental way? 2. The edit distance used is perfectly legitimated by the approach, and seems to provide good results. However, some other ways are open to the test, which I would have liked the author discuss. For instance, it is not clear whether the edit distance effectively reflects the closeness of the planning problem to the actual problem we want to solve. E.g. adding/deleting a precondition can modify totally the planning problem, even making it unsolvable (or requiring much more actions to reach the goal). A possible solution would be to run a planner on the (possibly relaxed) model being evaluated, in order to collect relevant data on the resolvability, or complexity, of the problem. The reason is that running a relaxed version of the problem (e.g. delete relaxation) is cheap. This evaluation/heuristic would provide a more domain dynamics oriented distance, and help to trace back to the planning process of the hidden model under the principle of rationality. On the practical side, the behavioural models of the (planning) agents are generally not available, and the problem specifications are generally poorly described and incomplete, and the different modelling languages in industry are not cross-compatible. I would not base my motivations for the present action model learning on this assumption, as stated in the Introduction. It is not clear to me, why deletes should be included in the model, as a delete relaxation version of the planning model would solve the problem, unless some observations are incompatible with a relaxed solution.

Review 2

Significance:	**2**: (modest contribution or average impact)
Soundness:	**3**: (correct)
Scholarship:	**2**: (relevant literature cited but could be expanded)
Clarity:	**3**: (well organized and well written)
Reproducibility:	**3**: (authors describe the implementation and domains in sufficient detail)
Overall evaluation:	**-1**: (weak reject)
Review:	The paper was a pleasure to read. However, as is the wont of most well-written papers, it has attracted a great many questions! I have ordered them by importance, please respond to them as space permits.

1) Model Recognition versus model learning? Though the problem is posed as “model recognition”, the problem definition is to arrive at a model from observed traces. Unless I am missing something, this is identical to the numerous attempts at model learning for planning. With full observed state-action pairs, the problem is simple (e.g. take the intersection of observed fluents to arrive at preconditions, etc.). But there are many approaches [ARMS; Yang et al. AIJ 2007] [RIM; Hankz IJCAI 2013] [AMAN; Hankz IJCAI 2013] [LOCM; Creswell et al. 2013] [LOCM+; Gregory et al. KEPS 2017] that deal with incomplete and even noisy traces and build a classical PDDL model out of it. What is the difference, if any?

There is a passing mention of this in the introduction, in the sense that models may not be built from scratch. That is hardly the case for the model learning work as well (e.g. RIM starts with a partial model). This is also true for the model reconciliation work [Sreedharan et al. ICAPS 2018] referred to in the paper whose starting point is an annotated PDDL which is equivalent to a set of models.

1a) Evaluations: Existing PDDL model learning techniques that deal with partial traces should then be baselines to compare against if the proposed approach is to be established.

1b) [Suggestion] Of course, with noisy observations a model may not exist as per the proposed method. The observation edits in [Sohrabi et al. IJCAI 2016] referred to in the paper can be handy then.

2) Model Recognition versus model reconciliation? The authors make an interesting connection to model reconciliation [Chakraborti et al. IJCAI 2018] in the end. However, I posit that the connection is much deeper than what the authors suspect. It is true that the model reconciliation work is done with a target model in mind. However, note that one of them may be empty (as in this paper). This was precisely the approach taken for the recent work on "visualizations an for explainable plan agentning" [Chakraborti et al. XAIP 2018] where the model reconciliation approach was instantiated with an empty model to find the minimal model that supports a given plan. The minimal complete solutions there correspond to the closest set.

Unless I am mistaken, the allowable changes here are inputs as well to the AJO-based approach, in addition to the empty model. That means, much like the two models in model reconciliation, we can conceive a full model here as

well with all possible domain conditions. Thus, the approach seems to be an instantiation of model reconciliation with an empty model and a "full model", an approach which has already been demonstrated with an empty model and an original model.

Furthermore, while this is not a problem for model reconciliation in that it solves a completely different problem of plan explanations, the need to specify possible changes does appear to be a severe limitation in the current context of model recognition. To put it in more grandiose terms (see 3 below), a model recognizer ideally needs the ability to "imagine".

In addition to the empty model variation, the current work also pretty much follows the model reconciliation approach with “edit functions” in model space. I recognize that the latter with the empty model approach has not been formalized yet, and I like the proposed formulation, but I am not sure it adds new insight other than the probability estimate (which is questionable; see 3b).

[Some more related work]

3) Model Updates in Story Telling: One application where domain edits are natural is in story telling (relevant to the second experimental setup on replanning on failure as well) using planning. Of particular relevance are systems such as ANTON [Porteous et al. AAMAS 2015] which can generate possibly relevant model updates to make. A model recognition algorithm should be similarly able to imagine up a domain to support an observed plan.

3a) More Model Updates: Similarly, in MARSHALL [Bryce et al. IJCAI 2016] authors tried to capture the drift of a planning model by means of a process very similar to the edit functions here that preserve STRIPS consistency -- there, the imagination task was to hypothesize possible drifts.

3b) Why Bernoulli process? As in the above works on story-telling and model drift which are likely driven by natural or higher-order processes that could and should be modeled, I fail to see why that is relevant in the model recognition setting proposed here. I would imagine a simple model like least edits would probably do the job of finding a closest consistent model.

4) In “Full observability of the executed plan” I don’t quite get why there would be a single model that would produce that trajectory. For example, a Gridworld where there is no

delete effect on the previous location while moving can support any plan from the normal Gridworld as well.

5) In “Partial observability of the executed plan”, (2) is a pretty big assumption that almost never holds (especially in navigation domains). I don’t think [Ramirez 2012] ever assumed a single optimal plan.

6) [Suggestion] Parameters: It seems to me that there may be interesting areas to explore in considering model abstractions in some of the transitions – e.g. by dropping parameters (for “predicate abstractions” in action dons). Definition 1 for comparability can be relaxed to achieve this.

Overall, I loved the formulation and presentation. However, the work lacks novelty when viewed in the light of several existing works (as mentioned above) and its significance is thus questionable. The problem is phrased as "model reconciliation" but I can't see any difference with traditional model learning approaches. If there isn't other than the compilation to classical planning, then that compilation must be put to test against those approaches. I would like to hear the authors opinions on the same before I can decide on a more informed score.

Typos/Minor:

Page-4: Instead, our approach is to estimate of P --> estimate P using an edit distance referred to --> edit distance defined for STRIPS planning models.
Page-5 top: we we
Page-5 2nd column: the edition of STRIPS planning model --> editing of a STRIPS planning model (and also other mentions of edition)

Bonus: A free challenge problem to consider –

Though automated planning has always taken pride in being the declarative way towards AI, it requires models whose specification process itself is procedural. That is to say, the domain writer has to perform all sorts of mental arithmetic (often ending up with unintended side effects in the domain) in order to figure out what action model would produce the kind of behaviors they want to see in an agent when all they want is the desired behavior. Imagine then a new paradigm for writing domain models where the domain writer provides examples of plans – e.g. “Here is a state, here is a plan, it achieves this goal.” – and the interface is able to do “model recognition” and come back with possible models that can support the requested behavior. And so on,

	iteratively, till the process converges to a single or a reasonable set of models. It is a possibility that is intriguing to me, especially given the related work of late that I mentioned above and might be of interest to the authors as well given their progress in very similar directions. The setting has parallels to teaching by demonstrations traditionally studied in continuous domains in robotics and ML. Even for that general setting the algorithm designer would have the opportunity to pick a representation that best serves their domain of operation, whereas this setup would be geared more towards the specific process of writing PDDL domains which remains inaccessible to non-planning people.

Review 3

Significance:	**2**: (modest contribution or average impact)
Soundness:	**2**: (minor inconsistencies or small fixable errors)
Scholarship:	**2**: (relevant literature cited but could be expanded) I think the paper would have benefited from a better understanding of why model recognition is needed in the light of existing literature in learning models.
Clarity:	**2**: (mostly readable with some room for improvement) There are some typos that the authors should
Reproducibility:	**5**: (code and domains (whichever apply) are already publicly available)
Overall evaluation:	**-1**: (weak reject)
Review:	The paper introduces the task of model recognition, which is a classification task in which, given an observation of a plan's execution and a set of model classes (or a set of incomplete models), the objective is to classify which of the models in the set is most likely the one that produces the output. Overall, the paper is reasonably well written. I believe the core problem is indeed interesting. I had a bit of a hard time making sense of the relevance of the theoretical model in this paper. First, the theoretical formulation is given in terms of a likelihood, together with a graphical model. This is interesting, however very quickly one realizes it is obvious that it is not computable (BTW, it is not clear in the paragraph immediately after the definitions whether or not the authors are saying that it is intractable or infinite and hence not computable). Then the paper proceeds to connect this probability-theory definition with an edit distance notion introduced in previous work. While this connection makes sense to me, I kept on wondering

	what was the relevance of the theoretical model. Although I do like theoretical models, I felt that in this paper it did not add too much, especially when we go to the experimental section and find that, instead of likelihoods, other notions, like accuracy, seem to be more informative. I could not reconcile these two notions (accuracy/likelihoods) in my mind and I just felt that the paper needed a little bit more work to present these concepts more clearly. Another aspect that I felt was weak in the experimental evaluation was the second part 'Recognizing failures'. I had a hard time making sense of this new concept ('failure') and relating this to the rest of the content. Perhaps, there is a bunch of nice tasks that can be mapped into model recognition that I have been missing and that the paper could have enumerated. Over all, I believe this is an interesting topic. I think this paper could improve significantly with a better motivation, a stronger argument in favor of the usefulness of the theoretical model, and a thorough identification of the tasks that can be captured by model recognition. More specific: - I thought it was kind of awkward that effects, both positive and negative, can be literals. While I can understand that a negated literal could be a positive effect, I believe that the most important problem is that, unless more assumptions are imposed, this makes the definition of \theta wrong given the definition of state that you've chosen. - The write up of the experimental section suggests that the first round of experiments (automata) had very impressive results. I would suggest a tone-down here since indeed the way you represent this problem should have produced those results.