DOI: 10.1109/ICRA40945.2020.9197477
URL: https://ieeexplore.ieee.org/document/9197477

Please cite as:
A. S. Bauer, P. Schmaus, F. Stulp and D. Leidner, "Probabilistic Effect Prediction through Semantic Augmentation and Physical Simulation," 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 2020, pp. 9278-9284, doi: 10.1109/ICRA40945.2020.9197477.

Probabilistic Effect Prediction through Semantic Augmentation and Physical Simulation

Adrian S. Bauer[1], Peter Schmaus[1], Freek Stulp[1], Daniel Leidner[1]

***Abstract*— Nowadays, robots are mechanically able to perform highly demanding tasks, where AI-based planning methods are used to schedule a sequence of actions that result in the desired effect. However, it is not always possible to know the exact outcome of an action in advance, as failure situations may occur at any time. To enhance failure tolerance, we propose to predict the effects of robot actions by augmenting collected experience with semantic knowledge and leveraging realistic physics simulations. That is, we consider semantic similarity of actions in order to predict outcome probabilities for previously unknown tasks. Furthermore, physical simulation is used to gather simulated experience that makes the approach robust even in extreme cases. We show how this concept is used to predict action success probabilities and how this information can be exploited throughout future planning trials. The concept is evaluated in a series of real world experiments conducted with the humanoid robot Rollin' Justin.**

I. Introduction

Big advances in mechanics and control led us to robots that are able to demonstrate impressive manipulation skills. Yet, the only service robots sharing space with humans are vacuum cleaners and lawn mowers with limited manipulation capabilities. One reason for this discrepancy is the lack of robust decision making capabilities, including the ability to handle failures and unforeseen effects. In scientific laboratories and demonstration areas this is often masked by tuning the environment according to the needs of the robots. In this environments, the action results are uniquely defined, making them behave like deterministic environments.

Deterministic environments allow to use the popular approach of integrated task and motion planning for planning and decision making [1], also referred to as hybrid reasoning. The core idea of hybrid reasoning is to generate a symbolic plan from symbolic action representations and translate it into a sequence of robot motions with matching geometrical effects. This approach is restricted to known environments and requires explicit symbolic representations of actions in form of deterministic state transitions. However, real world environments are not always deterministic, instead they are nearly always probabilistic.

In pop cultural context, robots are often displayed as embodied calculating machines. They are able to calculate success probabilities for long action sequences accounting for many external parameters such as C3PO in Star Wars V stating that *"the possibility of successfully navigating an asteroid field is approximately 3,720 to 1"*[2]. However, achieving even profound probabilistic effect estimation for robots, that utilize hybrid reasoning, requires all possible effects of actions and their respective probabilities to be specified explicitly.

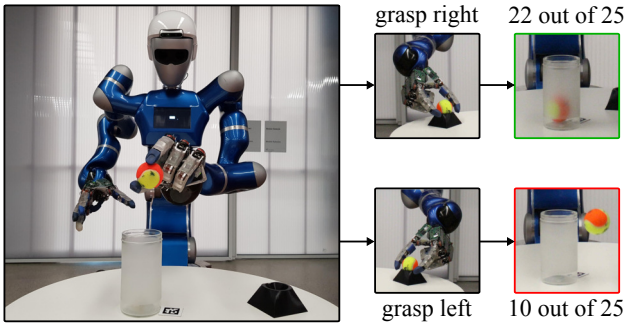


Fig. 1: Odds that Rollin' Justin is able to drop the ball into the container with the left or right hand.

Because of the high complexity of possible real world interactions, effect probabilities must not be hard coded, but have to be learned over time as visualized in Fig. 1. Moreover, once a robot learned for example that placing a glass on the edge of a table can lead to this glass falling down with a certain probability, it should be able to generalize this knowledge to other glasses. Existing work tackling the issue of learning probabilistic action representations from experience [3], [4], [5] is mostly concerned with extracting meaningful action effect representations from the data. We, on the other hand, assume that the possible effects of an action are accessible and aim at generalizing knowledge about their probabilities.

Our contributions in this paper include (i) a probabilistic extension to the concept of *action templates* [6] toward *probabilistic action templates*, (ii) a framework to derive effect probabilities from little experience and physical simulation that generalizes to abstract object classes, (iii) the adaptation of a probabilistic symbolic planner to optimize a sequence of real world robot actions w. r. t. success probabilities.

II. Related Work

This chapter gives an overview of work in related fields of this paper.

A. Probabilistic Action Representations

Failure tolerant planning can be achieved through explicit modeling of potential failure situations. The main idea of this paper is to explore failure probabilities over time rather than hard-coding them. This idea is inspired by the work of [7], where a symbolic manipulation representation is derived from sensorimotor data. In particular, the robot extends its

[1] German Aerospace Center (DLR), Institute of Robotics and Mechatronics, 82234 Weßling, Germany, Contact: adrian.bauer@dlr.de

action representation with newly observed effects. Probabilistic action representations can also be applied to fault tolerant planning, as it was showcased by [8]. The authors propose to represent possible failure situations as secondary effects of actions. As a result, the utilized planner is able to optimize the sequence of actions w. r. t. failure probabilities.

B. Hybrid Reasoning

In the early years of robotics, planning mostly considered symbolic deterministic domains. Algorithms such as the famous STRIPS solver [9] worked on a purely symbolic description of states and actions. Shakey [10], an example for an early robot using STRIPS, was even equipped with basic error recovery strategies. However, the real world is much more complex than symbolic representations can capture. Especially for generating feasible and collision-free robot motion paths, geometric considerations play an important role. That is why *hybrid reasoning* (or integrated task and motion planning) was firstly proposed as an interaction between symbolic and geometric planning by [11]. Their algorithm couples geometric configurations to symbolic representations in *accessibility lists*. Once a solution is found in the symbolic space, according trajectories are created by the motion planner. In case of errors, backtracking is employed to select different geometric configurations from the accessibility list that allow for collision-free paths.

This approach was further investigated for example by [12] who advocate for a strongly hierarchical planning algorithm treating geometric planning as the lowest hierarchical step. Furthermore, [13] show the integration of symbolic planning and motion planning on a real robotic platform. They highlight the need for geometric backtracking which allows to reconsider geometric choices in previous actions if they negatively impact a following action. Especially relevant for this work is the concept of action templates (ATs) that has been introduced by [14], [6]. ATs encode actions in a hybrid robot-agnostic way. The symbolic information is encoded in *Planning Domain Definition Language (PDDL)* [15] in the *symbolic header* of the AT, complemented by robot-agnostic geometric information in the *geometric body*. This design allows to use ATs in hybrid reasoning settings across multiple platforms. Since ATs are restricted to PDDL, they do not support probabilistic action effects.

C. Prospection and Mental Imagery

The role of mental imagery for humans has been investigated in the domain of neurosciences. [16] gives an overview of some related work and draws attention to the importance of mental imagery for tasks like perception or motor control. It has been argued that simulation could play a similarly important role in robotics for predicting action effects [17], [18]. The work by [19] is well known for this research direction. The authors focus on qualitative reasoning through "envisioning", which describes the process of inferring effects based on simulation-based projections. This enables the agent to predict probabilistic effects related to task parameterization.

D. Semantic State Inference

The symbolic planning process underlying the hybrid reasoning approach depends on a symbolic description of the world. Usually the symbolic world state is hard-coded for a scenario and updated only based on the described effects of actions executed by the robot. Keeping the world state up-to-date is especially challenging when effects of actions cannot be predicted exactly. This is the case when dealing with probabilistic action effects or in case of possible failures. [20] present a framework for synchronizing a physical simulation of the world with the real world state and extracting semantic representations from it. We use this framework for retrieving the semantic world state from simulations we run.

III. Representation of Probabilistic Action Templates

The skills evaluated in this paper have been programmed by experts. They are represented in a manner similar to what has been described as *Action Templates (ATs)* in [14]. As presented in Sec. II-B, ATs dissociate symbolic and geometric information of an action into the symbolic header and the geometric body.

The main difference of *Probabilistic Action Templates (PATs)* is that their symbolic header can contain probabilistic effects. These are stated in *Probabilistic PDDL (PPDDL)* [21]. Essentially they are a list of tuples containing each a probability and an associated effect. The symbolic definition for an example PAT is provided in listing 1.

```
1  :action _object.drop_over:
2    :parameters (?o - _object
3                 ?m - _manipulator
4                 ?c - _container)
5    :precondition (and(bound ?o ?m))
6    :effect (probabilistic
7                 0.7 (and(in ?o ?c)
8                         (not(bound ?o ?m)))
9                 0.3 (and(on floor ?o)
10                        (not(bound ?o ?m))))
```

Listing 1: Examplary semantic header of a "drop_over" PAT.

New effects are added to the PAT during the whole lifetime of the system. Thus, the list of possible effects might grow beyond large numbers, slowing down the process of planning. This is prevented by storing the full experience of action executions in a database while the PAT maintains only a reduced list of effects. As in [3], effects with a probability below a predefined threshold p_{sig} are subsumed as "noise" in the PAT. They happen so rarely that they need not be considered for planning. Ignoring them keeps the PATs from aggregating unnecessary information over time.

We do, however, restrict the use of PPDDL to a subset of what PPDDL allows. Our approach focuses on actions that do not have universal or existential effects. In other words, it has to be specified which objects are affected by an action.

IV. Predicting Effect Probabilities

The aim of this work is to predict effect probabilities as an agent is given a certain task and improving the prediction over time as the agent repeatedly encounters similar

scenarios. This is achieved in three consecutive steps. We begin by describing a naive baseline approach for estimating effect probabilities (see Sec. IV-A). On top of this it is desired to speed up the estimation process for new objects and new scenarios using generalized experience (see Sec. IV-B). Finally, we propose to employ a physical simulation to refine predicted effect probabilities by generating additional artificial experience (see Sec. IV-C). The process is visualized in Fig. 2.

A. Baseline Approach

The baseline approach for predicting the effect probabilities of an action is based on counting the times an effect e occurred ($\#e$) after execution of the action and its total execution count ($\#tot$). The probability $\hat{p}_{a,e}$ of encountering effect e after execution of a is:

$$\hat{p}_{a,e} = \frac{\#e}{\#tot} \tag{1}$$

A drawback of this approach clearly lies within its imprecision in case of few experience. Furthermore, the baseline approach does not allow to generalize from experience gained in similar scenarios.

B. Updating Effect Probabilities Online

In order to enable the robot to estimate effect probabilities from few examples, we exploit *similarity of actions* to speed-up effect probability estimation in a new situation. Similarity follows directly from the PATs. As its name suggests, a PAT is a *template* for actions from which concrete action instances can be derived by assigning a value to each of its parameters. Thus, an action is a combination of low level robot functions (such as motions) generated from a PAT by assigning objects to all parameters.

Listing 1 shows a PAT with three parameters *?o*, *?m*, and *?c* and the range of possible objects that they can be assigned to. *?o* depicts the most general case as it can be assigned any *_object*. *?m* and *?c* are more restrictive since they only accept objects of type *_manipulator* respectively *_container*.

For the baseline approach presented above, each action instance created from the PAT is interpreted as a separate action. Thus, even though the action *drop_over(ball, right_arm, bucket1)* might have been executed many times, the algorithm is unable to generalize its knowledge to a new action *drop_over(ball, left_arm, bucket2)*. However, it would be ill-advised to consider all actions instantiated from a common PAT as similar since some parameters can take on a wide variety of values. In listing 1, for example, *?o* can be assigned any object of the most generic type *_object*. Yet, it is not helpful to generalize from dropping paper into a basket to dropping pills into a small box.

Given a hierarchical object database as described in [6], we consider two actions to be similar if *(i)* they are generated from the same PAT and *(ii)* for each parameter the assigned objects for both actions share the same parent.

As an example, we assume two actions are generated from PAT *drop_over* by assigning *?o* = *ball_1* and *?o* = *football*. Furthermore, we assume for simplicity that the assignments for *?m* and *?c* are the same in both actions and that *ball_1* and *football* derive from the class *_ball* which derives from *_object*. This results in two similar actions. However, assigning *?o* = *bottle*, which derives from *_container*, deriving from *_object*, is not considered a similar action, since *bottle* and *ball_1* do not *directly* derive from the same class.

The definition of similarity is used to collect a set of all experience gained with similar actions. The following algorithm models a prior estimate of effect probabilities for the given action based on this set. The set of experience gained with all actions similar to the prototypical action a is referred to as $\mathcal{S}_a$.

We model the prior probability of an effect e of action a as a linear model that represents the predicted effect probability $\tilde{p}_{a,e}$ as the sum of the mean probability $\mu_{e,\mathcal{S}_a}$ of e in the set $\mathcal{S}_a$ and an additional factor for the impact $i_{as,e,\mathcal{S}}$ of each assignment $as = \{V_i = o_i\}$ of object o_i to parameter V_i in a:

$$\tilde{p}_{a,e} = \min\left(1, \max\left(0, \mu_{e,\mathcal{S}_a} + \sum_{as \in a} i_{as,e,\mathcal{S}_a}\right)\right) \tag{2}$$

The impact of an assignment in $\mathcal{S}_a$ is measured as the difference between the average probability of e in all actions that contain the parameter assignment as called $\mu_{e,\mathcal{S}_a,as}$ and the mean effect probability $\mu_{e,\mathcal{S}_a}$:

$$i_{as,e,\mathcal{S}_a} = \mu_{e,\mathcal{S}_a} - \mu_{e,\mathcal{S}_a,as} \tag{3}$$

The prior is used to generate a proper beta distribution $\mathrm{B}(\alpha, \beta)$ in order to describe the probability that $\tilde{p}_{a,e}$ adopts a certain value. This is necessary as the goal is to generate a posterior distribution over $p_{a,e}$ based on the prior and the experience gained by executing a. The likelihood $\mathcal{L}(\mathcal{D}_a | p_{a,e})$ of our experience (or data) $\mathcal{D}_a$ given a certain value for $p_{a,e}$ follows a binomial distribution for which the conjugate prior is a beta distribution. This is formalized as

$$p_{post}(p_{a,e}) \propto \mathcal{L}(\mathcal{D}_a | p_{a,e}) \cdot p_{prio}(p_{a,e}) \tag{4}$$

The prior beta distribution is specified by finding values for α and β that satisfy two conditions: Firstly, the expectation of the prior beta distribution has to be equal to $\tilde{p}_{a,e}$ from (2). This imposes the restriction

$$\mathbb{E}[\mathrm{B}(\alpha, \beta)] = \frac{\alpha}{\alpha + \beta} = \tilde{p}_{a,e} \tag{5}$$

Secondly, the sum of α and β must be $\alpha + \beta = 8$. Intuitively this sum is interpreted as how much we trust our prior. The higher the value, the more we trust our prior to be true. The value of 8 empirically proved to generate good results and means that we put the same trust in the prior and the data after 8 executions of a. Given α and β from the prior and the counts $\#e$ and $\#tot$ (compare Sec. IV-A) from the data, the posterior distribution is

$$p_{post}(p_{a,e}) \sim \mathrm{B}(\alpha + \#e, \beta + (\#tot - \#e)) \tag{6}$$

This distribution over the possible values of $p_{a,e}$ is now exploited to predict the effect probability of e for action a.

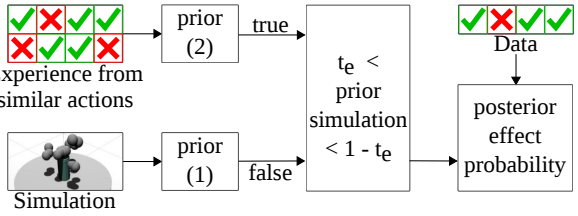


Fig. 2: Flowchart depicting the process of computing the posterior effect probability

For planning, we use the maximum likelihood estimate for $p_{a,e}$ which is analogue to (5)

$$p_{a,e,MLE} = \mathbb{E}\left[p_{post}\left(p_{a,e}\right)\right] = \frac{\alpha + \#e}{\alpha + \beta + \#tot} \tag{7}$$

Compared to the baseline approach, this prediction of the effect probability already performs well in the case of no or few experience with executing an action. Since the prior acts as "anchor" for the effect probability, it prevents the value from jumping, which is the case in the baseline approach. On the other hand, the anchor may also slow down adaption to extreme values, such as actions that are never (or always) successful. To improve the approach further in this regard, the next section integrates physical simulation as additional source of information.

C. Refining Predictions with Physical Simulation

Our approach to predict effect probabilities allows to generalize experience to new scenarios. This is done by employing experience collected in scenarios containing the same type of objects. However, not all objects behave the same just because they derive from a common class. An example is given w. r. t. listing 1:

The base assumption is that we are able to generalize experience collected when dropping balls over containers to other balls and other containers. However, in some scenarios this might not be the case, e. g. a football will never end up in a glass after dropping it over the latter. And since the probability for this effect is an extreme value (0 in this case), the prior generated according to (2) will always drag the posterior towards the mean.

This problem is avoided by employing physical simulation in terms of simulating an action and inferring the effect probabilities from simulation. As we believe that probabilistic effects in manipulation tasks are induced by epistemic uncertainty, we introduce small errors at each step of the action in simulation. The scaling of the errors must be adapted to reproduce the probabilities that are already known from experience, e. g. using an evolutionary algorithm as in [20].

We only trust the simulation if it predicts extreme values, i. e. values that are closer to 0 or 1 than a predefined threshold t_e, Formally a probability p is considered to be extreme if either $p \leq t_e$ or $p \geq (1 - t_e)$. In these cases the prior is created according to (5) based on the estimated probability of the simulation instead of the one from (2). The number of trials in the simulation depends on the selected threshold. We use a threshold of $t_e = 0.04$ and 25 trials in simulation.

Fig. 3: The containers used in the experiment. From left to right: cylinder, bread_box, bowl, glass.

TABLE I: Success rates for all parameter combinations given in percentage and absolute numbers

		_manipulator left_arm	_manipulator right_arm	total
_container	glass	5/25 20.0 %	10/25 40.0 %	15/50 30.0 %
	bread_box	11/25 44.0 %	24/25 96.0 %	35/50 70.0 %
	cylinder	10/25 40.0 %	22/25 88.0 %	32/50 64.0 %
	bowl	15/25 60.0 %	16/25 64.0 %	31/50 62.0 %
total		41/100 41.0 %	72/100 72.0 %	113/200 56.5 %

This means that the prior from experience is replaced with the prior from simulation if at most 1 simulated execution disagrees with all other executions.

V. Experiments

We evaluate our concept by conducting an experiment with the humanoid robot "Rollin' Justin" [22].

A. Experiment Setup

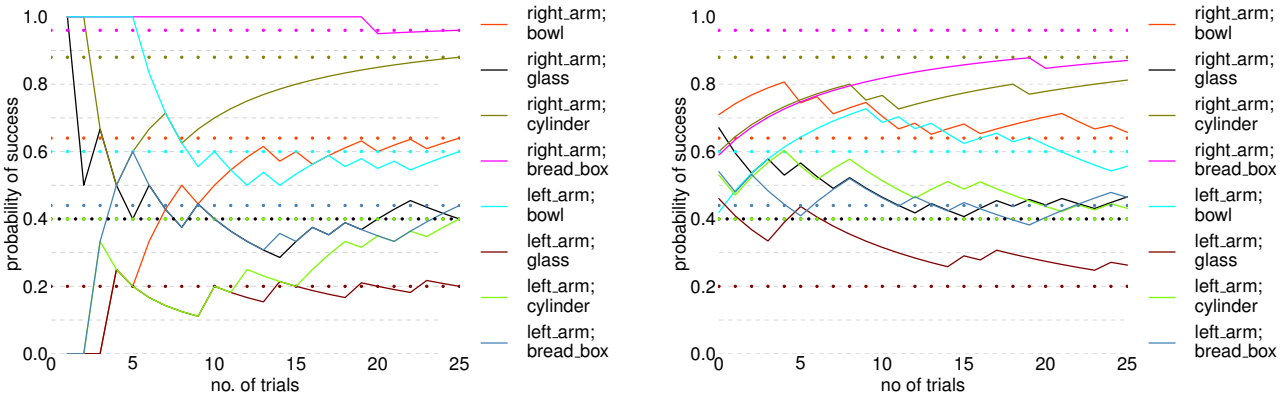
The overall task for Justin is to pick-up a tennis ball and drop it over one of four different containers. The ball and one of the containers are placed on a table in front of Justin as shown in Fig. 1.

The four containers are of different shape (see Fig. 3), resulting in different probabilities for the tennis ball to end up inside them after dropping it. The ball is placed on a stand to simplify grasping for the robot. Detection and localization of the ball stand and table are carried out with APRIL tags [23], the position of the container with respect to the table is fixed. We carry out 25 trials with each manipulator-container combination, resulting in a total of 200 trials.

A defect in the left manipulator, a DLR Hand II [24], is simulated by decreasing its joint stiffness, resulting in frequent failures as the ball is not reliably grasped. The trials are manually labeled as either "successful" if the ball ended up in the container or "failed" otherwise. Thus, "failed" collects all failure states such as *on(ball, table), on(ball, floor)* etc. For methods to circumvent manual labeling see [20].

B. Results

The results for each combination of parameters are displayed in Table I. The 95% Clopper-Pearson confidence intervals [25] for the probabilities after 25 trials are not displayed in the table but are each below $\pm 3\%$. Overall 56.5% of the trials were successful but one can observe a big



(a) (b)

Fig. 4: Evolution of the estimated probability (a) in the baseline algorithm and (b) in our algorithm. Dotted lines represent ground truth values.

variation over objects as well as over manipulators. Overall the expected difference between right arm and left arm is well visible, since we deliberately simulated a failure in the left hand. We can also observe that the glass was the most difficult to drop the ball into while on average the bread box was the easiest container.

VI. Evaluation

The experiments provide the means to compare the performance of our algorithm to the baseline algorithm. In the following we define an error function, give details about the evaluation, and show the results.

A. Quality Measure

Assessing the quality of the predicted probabilities, requires a quality measure in terms of an error function. The error function is supposed to measure the difference between predicted effect probabilities and the ground truth. The ground truth $g_{\mathcal{P}}$ for manipulator-container combination $\mathcal{P}$ is defined as the success probability measured after 25 executions of the action.

Ground truth values are shown in Table I. The compared algorithms predict a new probability each time the action is executed. This results in a series of $l = 25$ predicted probabilities $\hat{P}_{\mathcal{P}} = [\hat{p}_1, \dots \hat{p}_l]$ for each parameter combination $\mathcal{P}$. We define the error function as the mean squared error:

$$E\left(\hat{P}_{\mathcal{P}}\right) = \frac{1}{l}\sum_{i=1}^{l}\left(\hat{p}_i - g_{\mathcal{P}}\right)^2 \tag{8}$$

B. Evaluation Results

We evaluate our algorithm in terms of a leave-one-out crossvalidation. For each parameterization we assume that the remaining experience records are known and only the one investigated is new. We compute the prior according to (2) and iterate through the data collected in the experiment. The posterior is updated accordingly each time the parameterization of interest is encountered. The result is shown in Fig. 4b.

Fig. 4a shows the results of the baseline algorithm. After the first time an parameterization has been observed, the predicted success probability is either 0 or 1. While some parameterizations such as *(right_arm, bread_box)* are rather constant, others such as *(right_arm, glass)* vary significantly. The summed error of the baseline approach for all parameterizations according to (8) is 0.219.

The summed error for our algorithm in the experiment is 0.06 amounting to an error reduction of 72.6% compared to the baseline. The improvement is prominent for the parameter combinations *(right_arm, bowl)* and *(right_arm, glass)*. One dominant difference between Fig. 4a and Fig. 4b is their range on the x-axis. The predictions from the baseline approach start only after the parameterization has been observed at least once. In contrast, our approach generates probability estimates right away, even without any experience of the investigated parameterization.

A challenge for our algorithm becomes visible in parameterization *(right_arm, bread_box)*. The success probability for this parameterization is quite extreme with a value of 96%, and cannot be explained very well by the linear model. It seems that there is an interaction effect in the data which the algorithm cannot reproduce. Thus, it underestimates the probability.

C. Impact of Physical Simulation

The use of simulation is demonstrated in the same scenario but with a different container. This time the robot tries to drop the tennis ball over a shot glass. The shot glass is too small for the ball even to fit in, thus, this action never succeeds.

Since the robot never dropped any object over a shot glass, the prior it uses without simulation is simply the mean success probability of the manipulator it is using. As shown in Fig. 6, our approach without physics simulation starts with the mean value of the manipulator used and takes long to converge even approximately to the true probability of 0. By using the physical simulation, however, a perfect approximation is generated for the constructed scenario.

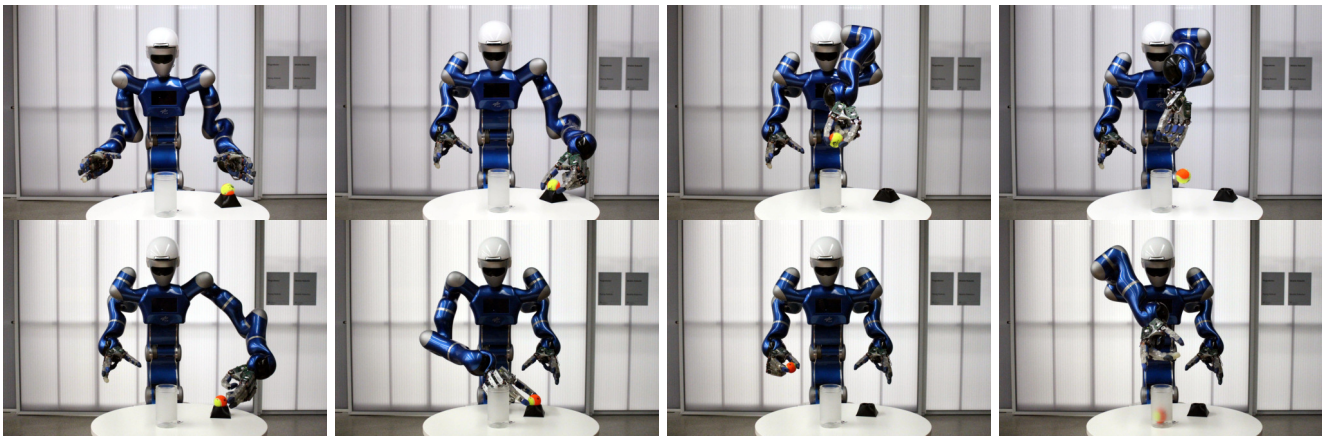


Fig. 5: Evaluation of the planner: The upper row depicts execution of the originally constructed plan, failing to reach the goal. The lower row shows the a more reliable plan generated by the probabilistic planner.

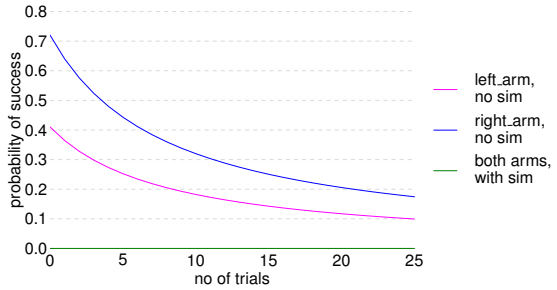


Fig. 6: Effect of using physical simulation in cases of extreme probabilities.

D. Planning with Estimated Probabilities

The effect probabilities are subsequently used to generate reliable plans. Therefore we adapt the fast downward planner (FD) [26] to cope with our PPDDL action descriptions. In a first step the probabilistic actions are determinized, i. e. a virtual action is created for each effect of the probabilistic action [27]. Afterwards the FD planner is used to find solutions to the deterministic problem. As soon as one solution is found, others are searched by iteratively deleting each of the actions in the solution one at a time from the set of available actions. By replanning, all possible solutions are found except those which are a mere permutation of any previously found one.

Next, the solutions are collapsed, i. e. solutions that are equal but contain different determinizations of an action are unified. Finally the success probability of a plan is computed by multiplying the effect probabilities of the relevant effects along the action chain.

We show how Justin can make use of the planner as it is confronted with the task to drop the ball over the cylinder where the ball is only reachable with the left hand. A new action *push_ball_right* moves the ball stand to a position from where it can be grasped with the right hand in 70% of cases. The probability for dropping the ball into the cylinder is 47% for the left hand and 80% for the right hand according to Fig. 4b.

From this, the planner generates two possible action sequences: *(i)* grasp the ball with the left hand and drop it in the cylinder with a success probability of 0.47 and *(ii)* push the ball with the left hand, grasp it with the right hand and drop it in the cylinder, yielding a success probability of $0.7 \cdot 0.8 = 0.56$. Fig. 5 shows how the robot is enabled to exploit the experience records to successfully drop the ball into the container with the second strategy (bottom), whereas it is likely to fail with the first sequence (top).

VII. Conclusion and Outlook

In this paper we proposed an approach for probabilistic effect prediction through semantic knowledge and physical simulation. The proposed algorithm outperforms a typical baseline approach as it exploits semantic similarity in a real world experiment. The definition of similarity is important to our approach. We used an hierarchical ontology and defined objects as similar if they derived from a common parent. Actions are considered similar if they are instantiated from a common PAT and use similar objects in their parameters.

Our approach treats similarity of objects as a binary label based on the hierarchical ontology. The hierarchy consists of object classes that proved to be meaningful throughout different scenarios we were working on. However, this is not always the case. Objects of the same class can potentially be very dissimilar and objects across classes may, nevertheless, be very similar. Thus, future work will focus on how to compute similarity relevant for the task at hand based on geometric features or more general ontologies to improve effect prediction toward failure tolerant robot planning.

Acknowledgment

This work is partly supported by the German Research Foundation (DFG) within the Collaborative Research Center EASE (SFB 1320) and by the Bavarian Ministry of Economic Affairs, Regional Development and Energy, within the projects ”SMiLE” (LABAY97) and ”SMiLE2gether” (LABAY102). The Authors would like to gratefully acknowledge the financial support and endorsement from the DLR Management Board Young Research Group Leader Program and the Executive Board Member for Space Research and Technology.

References

[1] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning. Theory & Practice*. Elsevier, May 2004.

[2] I. Kershner, "Star Wars: Episode V - The Empire Strikes Back," Jun. 1980.

[3] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning Symbolic Models of Stochastic Domains," *J. Artif. Intell. Res.*, vol. 29, pp. 309–352, Jul. 2007. doi: 10.1613/jair.2113

[4] L. S. Zettlemoyer, H. M. Pasula, and L. P. Kaelbling, "Learning Planning Rules in Noisy Stochastic Worlds," in *Proc. 20th Nat. Conf. Artificial Intelligence (AAAI)*, vol. 2. Pittsburgh, Pennsylvania: AAAI Press, Jul. 2005, pp. 911–918.

[5] D. Martínez, T. Ribeiro, K. Inoue, G. Alenyà Ribas, and C. Torras, "Learning probabilistic action models from interpretation transitions," in *Proc. 31st Int. Conf. Logic Programming (ICLP)*, Cork, Ireland, Aug. 2015, pp. 1–14.

[6] D. S. Leidner, *Cognitive Reasoning for Compliant Robot Manipulation*, 1st ed., ser. Springer Tracts in Advanced Robotics. Cham: Springer International Publishing, 2019, vol. 127.

[7] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning," *J. Artif. Intell. Res.*, vol. 61, pp. 215–289, 2018. doi: 10.1613/JAIR.5575

[8] R. M. Jensen, M. M. Veloso, and R. E. Bryant, "Fault Tolerant Planning: Toward Probabilistic Uncertainty Models in Symbolic Non-Deterministic Planning," in *Proc. 14th Int. Conf. Automated Planning and Scheduling (ICAPS)*, Whistler, Canada, 2004, pp. 335–344.

[9] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, Dec. 1971. doi: 10.1016/0004-3702(71)90010-5

[10] N. J. Nilsson, "Shakey the Robot," SRI International, Menlo Park, CA, Technical Note 323, Apr. 1984.

[11] F. Gravot, S. Cambon, and R. Alami, "aSyMov: A Planner That Deals with Intricate Symbolic and Geometric Problems," in *Robotics Research. The Eleventh International Symposium*, B. Siciliano, O. Khatib, P. Dario, and R. Chatila, Eds., vol. 15. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 100–110. doi: 10.1007/11008941_11

[12] L. P. Kaelbling and T. Lozano-Perez, "Hierarchical task and motion planning in the now," in *Proc. 2011 IEEE Int. Conf. Robotics and Automation (ICRA)*, Shanghai, China, May 2011, pp. 1470–1477. doi: 10.1109/ICRA.2011.5980391

[13] L. Karlsson, J. Bidot, F. Lagriffoul, A. Saffiotti, U. Hillenbrand, and F. Schmidt, "Combining task and path planning for a humanoid twoarm robotic system," *ICAPS Workshop Comb. Task Motion Plan. Real-World Appl.*, pp. 13–20, 2012.

[14] D. Leidner, C. Borst, and G. Hirzinger, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *Proc. 2012 IEEE-RAS Int. Conf. Humanoid Robots*, Nov. 2012, pp. 429–435. doi: 10.1109/HUMANOIDS.2012.6651555

[15] M. Ghallab, A. Howe, D. Christianson, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL - The Planning Domain Definition Language," *Fourth Int Artif. Intell. Plan. Syst. AIPS98 Plan. Comm.*, vol. 78, no. 4, pp. 1–27, Aug. 1998.

[16] S. M. Kosslyn, G. Ganis, and W. L. Thompson, "Neural foundations of imagery," *Nat. Rev. Neurosci.*, vol. 2, no. 9, pp. 635–642, Sep. 2001. doi: 10.1038/35090055

[17] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding," *Proc. Natl. Acad. Sci.*, vol. 110, no. 45, pp. 18 327–18 332, 2013. doi: 10.1073/PNAS.1306572110

[18] K. K. Szpunar, R. N. Spreng, and D. L. Schacter, "A taxonomy of prospection: Introducing an organizational framework for future-oriented cognition," *Proc. Natl. Acad. Sci.*, vol. 111, no. 52, pp. 18 414–18 421, Dec. 2014. doi: 10.1073/PNAS.1417144111

[19] L. Kunze and M. Beetz, "Envisioning the qualitative effects of robot manipulation actions using simulation-based projections," *Artificial Intelligence*, vol. 247, pp. 352–380, Jun. 2017. doi: 10.1016/J.ARTINT.2014.12.004

[20] A. S. Bauer, P. Schmaus, A. Albu-Schäffer, and D. Leidner, "Inferring Semantic State Transitions During Telerobotic Manipulation," in *Proc. 2018 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018, pp. 5517–5524. doi: 10.1109/IROS.2018.8594458

[21] H. L. S. Younes and M. L. Littman, "PPDDL1.0: An Extension to PDDL for Expressing Planning Domains with Probabilistic Effects," School of Computer Science Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-04-167, Oct. 2004.

[22] C. Borst, T. Wimbock, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P. R. Giordano, R. Konietschke, W. Sepp, S. Fuchs, C. Rink, A. Albu-Schaffer, and G. Hirzinger, "Rollin' Justin - Mobile platform with variable base," in *Proc. 2009 IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2009, pp. 1597–1598. doi: 10.1109/ROBOT.2009.5152586

[23] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. 2011 IEEE Int. Conf. Robotics and Automation (ICRA)*, Shanghai, China, May 2011, pp. 3400–3407. doi: 10.1109/ICRA.2011.5979561

[24] J. Butterfass, M. Grebenstein, H. Liu, and G. Hirzinger, "DLR-Hand II: Next Generation of a Dextrous Robot Hand," in *Proc. 2001 IEEE Int. Conf. Robotics and Automation (ICRA)*, vol. I, Seoul, Korea, May 2001, pp. 109–114. doi: 10.1109/ROBOT.2001.932538

[25] C. J. Clopper and E. S. Pearson, "The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial," *Biometrika*, vol. 26, no. 4, pp. 404–413, 1934. doi: 10.2307/2331986

[26] M. Helmert, "The Fast Downward Planning System," *J. Artif. Intell. Res.*, vol. 26, pp. 191–246, Jul. 2006. doi: 10.1613/JAIR.1705

[27] S. Yoon, A. Fern, R. Givan, and S. Kambhampati, "Probabilistic Planning via Determinization in Hindsight," in *Proc. 23rd AAAI Conf. Artificial Intelligence*. Chicago, Illinois, USA: AAAI Press, 2008, pp. 1010–1016.