©2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

SLOTH – the Interactive Workout Planner

Gregor Behnke[1], Florian Nielsen[2], Marvin Schiller[1], Pascal Bercher[1], Matthias Kraus[2], Wolfgang Minker[2], Birte Glimm[1], and Susanne Biundo[1]
[1] *Institute of Artificial Intelligence* [2] *Institute of Communications Engineering*
Ulm University, Ulm, Germany

***Abstract*—We present the mixed-initiative planning system SLOTH, which is designed to assist users in planning a fitness workout. Mixed-initiative planning systems are especially useful for *companion systems*, as they allow the seamless integration of the complex cognitive ability of planning into ambient assistance systems. This is achieved by integrating the user directly into the process of plan generation and thereby allowing him to specify these objectives and to be assisted in generating a plan that not only achieves his objectives, but at the same time also fits his preferences. We present the capabilities that are integrated into SLOTH and discuss the design choices and considerata that have to be taken into account when constructing a mixed-initiative planning system.**

1. Introduction

Traditionally, planning-based assistance systems behave in a black-box manner. The user describes a problem he has to solve. The planner then computes a sequence of actions which, if executed, solves the problem posed by the user. These instructions have to be followed without any (or only limited) possibility of deviation or personalisation. In some situations, such assistance is acceptable, e.g., if the user only desires to solve a problem in which he does not have any personal stakes. An example for such a situation is the setting-up of a home-theatre [1], [2].

If the user has personal interests and preferences, this black box interaction scheme can lead to significant problems in the human-companion relationship. This would render them useless for *companion systems*, which must be able to adapt their assistance to their individual users [3]. To mitigate this problem, the idea of mixed-initiative planning (MIP) was developed. It aims at integrating the user directly into the process of creating a plan, enabling him to exercise control on the collaboratively developed plan. As a result, the planner will provide a plan that is acceptable to the user. MIP provides the individualised assistance thought for in *companion systems* [3], as it allows the user to express these wishes and preferences to the system, which in turn adapts its assistance accordingly.

Only a few MIP systems have been studied – and implemented – before. In this paper, we present the new mixed-initiative planner SLOTH (the "Singularly Loveable and Obliging Training Helper"), which can assist human users in planning workouts for their fitness-training objectives. In this domain, most users have individual preferences (e.g. liking or disliking certain exercises or combinations thereof) making it an interesting application domain for a mixed-initiative planner. SLOTH extends the capabilities of previous mixed-initiative planners in several ways. First, our planner is thoroughly knowledge-based. We use a knowledge base containing necessary background knowledge of our application domain, which is used both to ensure internal model consistency, but also to enable the user to pose questions only indirectly related to the current planning process. Second, our planner uses a hybrid planning formalism which is able to capture the processes occurring during planning, i.e., both hierarchical and causal reasoning. Third, previous planners did not integrate the user into the plan-generation process itself, but rather used a critique-and-adapt cycle, where a solution was repeatedly presented and the user was able to request changes to it. Our planner integrates the user directly into the planning process, increasing its flexibility. Fourth, our planner uses advanced dialogue-techniques which take the user's mental capacity into account, such that it will neither be bored nor confused. Fifth, in contrast to most existing approaches to MIP, we have chosen a domain-independent approach, i.e., our planner can be used in any application domain. In the remainder of the paper, we will present the technical details of SLOTH, discuss the challenges arising when integrating users into a planning process, and show how SLOTH solves them.

2. Preliminaries

The main purpose of SLOTH is to provide adaptive support when developing a fitness workout. The assistance relies on the execution of automatically generated plans of action. For generating these plans, we rely on the planner PANDA [4], which generates knowledge-rich plans in a hierarchical manner. It relies on a hybrid planning framework [5], [6] that combines hierarchical task network (HTN) planning [7] with partial-order causal-link planning [8]. Hybrid planning seems well-suited for the endeavour of providing user-support for many reasons [9], most importantly because humans often solve their problems in a hierarchical manner [10] while also relying on causal reasoning.

In hybrid (and hierarchical) planning, plans are partially ordered sets of tasks. Tasks are 2-tuples, specifying their preconditions and effects – both being conjunctions of literals. There are two kinds of tasks: the primitive ones (also called actions) can be directly executed, whereas the abstract ones need to be further refined into courses of (more) primitive tasks. The different levels of abstraction are given implicitly by the task hierarchy: for every abstract task, the model contains a set of so-called decomposition methods. A decomposition method is simply a 2-tuple mapping an abstract task to a plan, which is a pre-defined standard solution to the respective abstract task [6]. This plan can in turn contain abstract tasks. Replacing an abstract task by the plan specified by one of its decomposition methods is called decomposing. Planning is done by successively decomposing abstract tasks until a plan is generated containing only primitive tasks, such that they can be arranged in an order that allows the plan to be executed [4], [6].

Knowledge-representation in SLOTH is based on an ontology, which is a suitable representation formalism to express concept hierarchies and relationships between concepts in a domain of interest. These relationships are formulated in the ontology language OWL,[1] which uses description logic (DL) as its theoretical underpinning. In SLOTH, the employed fragment of OWL corresponds to the DL $\mathcal{ALC}$ [11]. Concept hierarchies are expressed by subclass-superclass relationships (also called subsumption). Further relationships between concepts are formulated using existential and universal quantifiers and roles, which are interpreted as binary relations. Additionally, the language allows the expression of negation, equivalence and disjointness of concepts. To determine whether a subsumption between two concepts follows from the knowledge represented in an ontology, efficient automated reasoners can be used.

3. Related Work

Since its inception, AI planning has been used in a wide variety of application scenarios. Naturally, there have been situations where the use of a bare planning system has not been appropriate and MIP systems have been developed. Unfortunately, there has (as of now) not been much coherent work on developing such systems – neither in theory nor practice. Existing work in MIP has traditionally focussed on developing systems for a single application. The first such system was the planner developed in the TRAINS and TRIPS projects [12], which assisted a user in planning railway transport routes. This system was specifically tailored to that scenario, so much so, that it did not even use a general-purpose planner, but a dedicated solver. This contradicts the idea of planning to be a means of general (i.e. domain-independent) problem solving. SLOTH on the other hand uses a general-purpose planner, enabling it to be used in many different application domains.

One of the best-known mixed-initiative planners is MAPGEN, which was developed by NASA to aid in the action planning for the Mars rovers and was later also deployed in other deep-space missions [13], [14]. Like TRAINS/TRIPS, it has a specialised internal solver designed to cope with both time as resource constraints, as well as with competing needs from several groups of scientists. The overall aim of the planner is however not to provide an individualised plan, but to be able to mediate between the (usually incompatible) interests of several users.

In two other application scenarios, the developers of mixed-initiative planners have noted that classical (i.e. non-hierarchical) planning is not appropriate to model the constraints of the application, but that hierarchical planning provides for a flexible means to model them. These scenarios include generating action plans for firefighters tackling forest-fires (SIADEX [15]) and attack plans for military special operations groups (PASSAT [16]). All these planning systems are only capable of interacting with users based on completely developed plans. Their control-loop requires the planner to find a valid plan which is then presented to the user, who is asked for his opinion or advice. His response is then taken into account and planning starts again.

4. The SLOTH system

In SLOTH's application scenario, it is the user's objective to select a set of exercises for the current training session, which help him to reach a fitness objective (e.g. increasing strength or increasing stamina). In the planning model, these goals are modelled as abstract tasks, while the exercises themselves are primitive actions. Between these two, there are two levels of abstraction: training and workout. Workouts are predefined sets of exercises belonging together, which we gathered from fitness-training websites. A training describes a group of workouts achieving a common training objective, e.g., `FullBodyTraining`, which represents a training targeting the whole body, or `StrengthTraining`. These in turn can be decomposed into one or more workouts and those into primitive tasks. The remainder of this section introduces all the components of SLOTH. We start by describing the knowledge model and how it is used, then focus on planner-specific considerations and describe how change-requests are handled. Next, we describe how the planner interacts with the dialogue-management and it in turn with the user. Lastly, we show how explanations are integrated into the planner.

4.1. Overview

To assist the user, SLOTH guides him through the process of generating a plan in the fitness training domain. The user is repeatedly shown a current (partial) plan to consider. In the process, he is offered options to refine the current plan, which are based on the decomposition methods applicable to the abstract tasks the plan contains. The user is also given the opportunity to apply changes to the plan generated so far (cf. Fig. 1). These requests are handled by the dialogue management component (see Section 4.5), which also determines how these options are presented to the user. As

1. https://www.w3.org/TR/owl2-overview/

a result, the dialogue manager informs the planner of the user's reaction (or any other input, in case of a request to change the plan). The planner in turn reacts upon this input and selects a new plan for presentation to the user. This control loop repeats until the current plan is a solution to the planning problem and the user does not want to change it any further. In addition to this loop, the dialogue manager can also handle requests of the user for an explanation of the refinement options he is presented. These inquiries are not handled by the planner, but by a separate explanation component taking advantage of the fact the planning domain is (partially) constructed from an ontology (see Section 4.6).

4.2. Knowledge models

Each component of a mixed-initiative planner needs a dedicated domain model for the specific task, which that component has to perform. These distributed models naturally create consistency and redundancy issues. Recent work provides a method for utilising a central ontology, such that all necessary models can be derived from it automatically, ensuring consistency between the models without any redundancy [17]. We have applied this technique in SLOTH. As detailed by Behnke et at. [17], this technique not only ensures consistent models, but also provides additional benefits for cognitive systems. E.g. (at least some) decomposition methods in the planning model can be automatically derived from factual reasoning based on background knowledge stored in the ontology. In the case of our fitness-training domain, these methods, e.g., include those relating training objectives to the workouts which fulfil their objective. The planning domain specifies that every muscle group in the body has to be trained to constitute a `FullBodyTraining`. By using ontology reasoning, we can infer that a given set of workouts combined achieves this constraint and can thus create a decomposition method from `FullBodyTraining` into these workouts. In fact, all methods relating training objectives and workouts in our model are inferred fully automatically. As these decompositions are derived from background knowledge, we can also provide factual explanations for them, where previous approaches were not able to do so (see Section 4.6). To allow for the maximum degree of freedom during planning, we can also create decomposition methods based on the ontological descriptions of abstract tasks, if certain criteria are met. If the task is defined in terms of a conjunction of necessary concepts each describing a set of (sub-)tasks, we can create a decomposition method containing tasks each representing one of the these concepts. Decomposition methods for these tasks can again be created automatically.

4.3. Planning

The first step in assisting the user is to determine the objective which he actually wants to achieve. As the initial plan effectively describes the objective (or goal) of the user, prior to the planning process itself, we use a dialogue to determine the user's goal (see Section 4.5). We will discuss

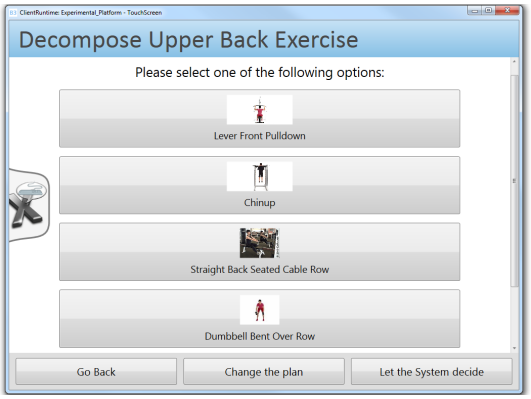

Figure 1. Selection dialogue in SLOTH

the design choices that were made when developing the planner that SLOTH uses, which adverse consequences the planner's strategy can have, and how to mend them.

As (almost) all other techniques developed for MIP, SLOTH presents the user with a current plan and inquires about the user's accord until a solution to which the user agrees has been reached. But in contrast to these techniques, a plan can be shared with the user while still in an incomplete or abstract stage, which allows the user to influence the planning process early on. This very general description of the planner's behaviour makes two design-choices of a mixed-initiative planner obvious: Which plan(s) should be presented to the user? and What the user should be asked about that plan? As we are using the planning formalism of *hybrid planning*, a natural answer to the latter question is to present the user with all the possible options to refine the current plan and to let him choose the one he wants, or let him indicate any preference between them. Then the former question boils down to deciding which planning algorithm to use, as it determines an order in which the plans are presented to the user. The main objective here is not to choose an efficient algorithm, but to not overexert the user's mental capacity. We chose a depth-first-search (DFS) strategy, which keeps the difference between two subsequent plans as local as possible. This enables the user to focus on the changes he wants to perform [18].

Despite the fact that depth-first strategies can be easily understood, they can make planning an extremely tedious process. This becomes apparent whenever – through the choices of the human user – the current plan cannot be refined into a solution any more. When using DFS, we here have to explore the complete search space below this point, often with repeatedly going through the same futile options.

To counteract this problem, we have added two capabilities to SLOTH. First, in an ideal world we would be able to construct the whole search space in advance and only present options to the user which can potentially lead to a solution. This however is not feasible, both for practical and for theoretical reasons. In general the search space of a planning problem is not finite as a planning problem can have infinitely many solutions, and even if

the search space is finite in practice it is often too large to be constructed explicitly, which is the very reason complex planning algorithms were developed. Also it would be unacceptable for a *companion system* to require a preparation phase of a few minutes before being able to assist the user by means of planning. Even though we take advantage of the fact that whenever SLOTH shows a plan to the user and provides options, the user will require some time (at least a few seconds) to understand the plan, to assess the presented options, and to make decisions. Every time a plan is presented to the user, the planner would ordinarily be idle. Instead SLOTH uses this time to filter out presented options which cannot lead to a solution. This is done via search in the plan space induced by each of the options. If we find a solution in one of them, we know that this option is still "viable", i.e., it leads to a solution. If in contrast the planner cannot find a solution below an option, we disable the respective option. Since the planner might need a few seconds (or even longer) to determine either outcome, we have to cope with the situation where we have presented the user with an option, he has (potentially) considered it, and we only then disable it. We will elaborate on strategies of how to handle this in Section 4.5. Replanning for each option is not always necessary, as we will usually have searched through (parts) of the search space below one of the current options in a previous run. We use a simple caching strategy to reuse results (i.e. the found solutions and constructed search spaces) in this scenario.

As a second means to overcome the disadvantages of DFS, we enable the user to not only react to the choices put by the planner before him, but also to pro-actively instruct the planner to alter the current plan in a specific way. By that, the user can "escape" parts of the search space which either do not contain solutions or revert previous decisions (or consequences thereof) he does not like. To allow this, SLOTH can handle so-called change requests, which are instructions of the user to alter the current plan in a specific way. We will elaborate on how these requests look like and how SLOTH reacts to them in the next section.

4.4. Change requests

Being able to react to spontaneous instructions from the user is one of the most important abilities for a MIP system. Without it, the system would not interact with the user, but the user would merely answer to questions without any change in initiative. In previous work, possible requests to change a plan were already investigated theoretically [19] and five types of requests and three different sets of constraints the planner has to adhere to when answering them were defined. These constraints express by how much the planner can deviate from the current plan: it either can only implement the changes requested by the user or also perform implied changes to ensure that the resulting changed plan is or can still be refined into a solution.

As the practical implementation of these requests has not been studied so far, we have concentrated on the most relevant type of request in the setting of fitness training: replace requests. These requests ask to replace one of the actions in the current plan with another one. Similarly to the situation of dead-ends, it should be possible to refine the plan resulting from this change request into a solution to not frustrate the user. To ensure this property, the planner might have to perform additional changes to the plan. Behnke et al. [19] have given three different levels of allowed additional changes: none at all, up to K, or any arbitrary change. As this essentially constitutes a single scale[2], we have to solve an optimisation problem: Find the plan with the least amount of alterations that fulfils the posed requests and can be refined into a solution. SLOTH solves this problem via local search. Despite the fact that this technique is designed to solve replace requests, it can be easily adapted to add, remove, and change order requests. We first perform the requested change itself and then perform a local breath first search in the plan space, thus ensuring that we will always find the plan with the minimal amount of alterations first. To test whether a plan is a solution, we use the cached results from the plan space exploration, detailed in the previous section, or if such a result is not available start a planning process from the current plan. This process will typically terminate quickly, as users tend to only pose change requests when the plan has already been refined several times, thus reducing the possible search space below that plan.

In addition to the ability to perform change requests we also have to keep track of previous requests. For example, if the user has requested to exchange a `BarbellSquat` with a `BodyWeightSquat` and subsequently asks to change another action, we should not change the `BodyWeightSquat` back into a `BarbellSquat`. SLOTH keeps track of past change requests, and checks them as "must contain" action constraints during local search. Here, the user might arrive at a situation where it becomes impossible to find a plan satisfying all past change requests (e.g. if two actions have to be included in the plan that exclude each other according to the training rules). In this case, we present the current list of constraints to the user and ask him which of them we are allowed to drop.

4.5. Dialogue Management

The dialogue management (DM) module of SLOTH addresses three challenges of MIP [18]: determining the initial planning task in a user-friendly dialogue, integrating the user into the planning process to provide for a collaborative decision making, and handling of failures during planning.

In order to generate a valid planning problem, the semantics of the dialogue and planning domain have to be coherent. Therefore, the DM module and the planner use a *mutual knowledge base* (see Section 4.2). This allows the dialogue to be structured in a hierarchical way, which helps the user to define an initial set of tasks (i.e. the tasks he wants achieved) efficiently. For example, the user states the goal that he would do *strength training in the upper body,*

2. The distinction is only interesting for a theoretical investigation of change requests.

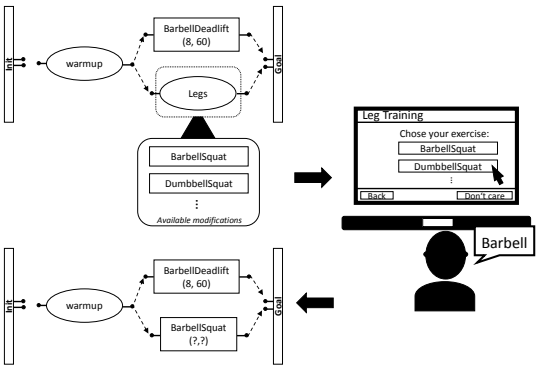

Figure 2. Integration of the user in the planning process

while working out two days a week. This goal definition is then used as the topmost element of the dialogue hierarchy, as the `StrengthTraining` can be decomposed into a set of workouts, whereas each workout consists of a set of exercises. In order to render the interaction user-friendly, the DM needs to handle free and complex naturally spoken utterances. For this purpose, a more sophisticated approach for linguistic analysis than generic grammars has to be applied. We use the statistically-driven approach of *language understanding intelligent service* (LUIS) [20], which allows to handle complex and natural user input, e.g., so-called "one-shot" utterances like, *"I want to do strength training for my upper body, two times a week"*, which then can be passed on directly to the planner, instead of conducting a lengthy dialogue. LUIS relies on the concepts of intents (what does the user want?) and entities (which parts of the utterance are interesting for the system?). The LUIS model in SLOTH has been trained on three different kinds of intents by the user: `Inform` the system about certain entities, `Request` the system for additional information on entities. `Confirm` previously provided information that the system has understood with a low confidence. Furthermore, the model has been trained on a set of entities. For the sentence, *"I want to do strength training, two times a week"*, "strength training" would be recognized as value for the entity `AbstractGoal` and "two times a week" as value for the entity `TrainingDates`. The intents and entities of an utterance are then transformed into a dialogue act of the following form:

$$\underbrace{Inform}_{Intent}(\underbrace{AbstractGoal}_{Slot} = \underbrace{"strength\ training"}_{Value})$$

As described in Section 4.3, the plan itself is presented to the user together with a list possible refinements. This leads to another critical issue in MIP regarding *if, when*, and *how* user-involvement during the planning process is generally useful or necessary, as this largely affects how the interaction with the planner is perceived by the user [21]. Therefore the DM component contains an elaborate decision model, integrating various information sources. This includes, e.g., the dialogue history, the kind of presented refinements, the user profile, and the current situation. The decision model can either initiate a user interaction, determine by itself that the planner should make the decision, or ask the user explicitly or implicitly for confirmation that the system decides on the next planning step. In case of user involvement the information on the current plan decision has to be communicated to the user (see Fig. 2). This means that the open decision and the corresponding choice between available modifications have to be represented in a suitable dialogue. Hence, the corresponding actions or methods need to be mapped to human-understandable dialogue information, using the *mutual knowledge base*. As the user is dependent on the reasoning capabilities of the planner, there may occur a mismatch between the user-expected plan and the system-generated one. In order to detect these critical situations in HCI, which may cause a loss of trust in the system behaviour, and to react appropriately, the DM component of SLOTH augments the dialogue by integrating additional content, e.g. explanation steps, and decides on the form of representation of selection or confirmation dialogue steps. This is conducted by using a POMDP-decision module [22]. This module monitors whether the users trust is endangered. At run-time the next action of the planned dialogue is compared to the one determined by the POMDP. If the next action is not the same as the POMDP would expect, the flow of the dialogue is interrupted, and the ongoing dialogue is augmented as mentioned above.

In planning, a failure occurs if the current plan is proved unsolvable (see Section 4.3). In case of a failure, DFS typically uses *backtracking*, i.e., rolling back planning steps, until the faulty decision is reverted. As this is a very tedious and frustrating process, *backtracking* may result in the user deeming the system's strategy naive and the system itself incompetent [18]. In order to prevent such a negative experience, the computing power of the planner can be used to determine whether a decision option for refinement is a so-called *dead-end* (i.e. leads only to faulty plan, see Section 4.3). The objective of the DM module is to convey this information appropriately to the user. In SLOTH this problem is addressed by providing explanations, which aim to increase the user's perceived system transparency [18]. In case of *backtracking*, the system informs the user that previously user-made decisions cannot lead to a solution and it explains to the user why the already made decisions have to be rolled back. *Dead-ends* are communicated by notifying the user that some refinement options had to be deactivated, because they would not lead to a solution.

4.6. Explaining Facts and Decompositions

The ontology used by SLOTH incorporates the training goals, training patterns, and exercises that can be used in interactive planning. Axioms specify which logical relationships hold in the domain, for instance, which muscles are trained by a particular exercise and where they are located in the body. This information is not only used in the process of planning, but is also used to inform the user about the properties of exercises, training objectives, the human anatomy, and how they are related, to foster the understanding of the

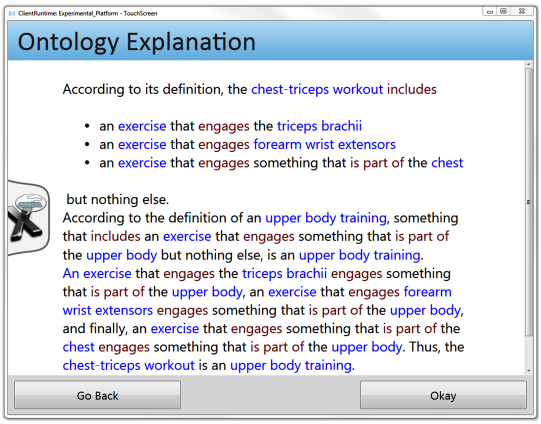

Figure 3. Generated explanation for "a chest-triceps workout is an upper body training" in SLOTH

available choices offered by MIP. In the approach chosen for SLOTH, decomposition methods available to planning always correspond to subsumption relationships in the ontology. For instance, using an automated ontology reasoner (SLOTH uses FaCT++,[3] but other choices are possible), the concept of `ChestTricepsWorkout` is derived to be a sub-concept of `UpperBodyTraining` based on the properties of the exercises specified as parts of the workout. So, for instance, if `ChestTricepsWorkout` includes (among others) an exercise for the `TricepsBrachii`, which according to the ontology is a part of the `UpperBody`, this exercise is inferred to be engaging the `UpperBody`. The concept of `UpperBodyTraining` is defined in the ontology as consisting of (only) exercises for training the upper body, which can be shown for `ChestTricepsWorkout`.

The explanation generation mechanism[4] employed by SLOTH uses a rule-based reasoning system (cf. [17]) to generate step-wise formal proofs for the derived subsumptions. These represent the underlying reasons for relationships between concepts in the domain, based on facts in the ontology, for example why `UpperBodyTraining` can be decomposed into `ChestTricepsWorkout`. Natural-language explanations, as shown in Fig. 3 for the example, are generated by applying text patterns to inference rules and description logic formulas in the proof (cf. [23]).

5. Conclusion

We have described the MIP system SLOTH, which assists users in planning their fitness training workouts. We described which subsystems are necessary for such a planner and how they can and should work together. We have further described the challenges that we faced and solutions we developed when developing the individual components. Especially, we have described new planning techniques (e.g. for answering change requests) that are useful for the development of future MIP systems.

3. http://owl.man.ac.uk/factplusplus/

4. http://verbalizer.github.io/

Acknowledgments

We acknowledge the support of the Transregional Collaborative Research Center SFB/TRR 62 "A Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

[1] P. Bercher, S. Biundo, T. Geier, T. Hörnle, F. Nothdurft, F. Richter, and B. Schattenberg, "Plan, repair, execute, explain – How planning helps to assemble your home theater," in *Proc. of ICAPS*, 2014.

[2] P. Bercher, F. Richter, T. Hörnle, T. Geier, D. Höller, G. Behnke, F. Nothdurft, F. Honold, W. Minker, M. Weber, and S. Biundo, "A planning-based assistance system for setting up a home theater," in *Proc. of AAAI*, 2015.

[3] S. Biundo and A. Wendemuth, "*Companion*-technology for cognitive technical systems," *Künstliche Intelligenz*, no. 1, 2016, Special Issue on Companion Technologies.

[4] P. Bercher, S. Keen, and S. Biundo, "Hybrid planning heuristics based on task decomposition graphs," in *Proc. of SoCS*, 2014.

[5] S. Biundo and B. Schattenberg, "From abstract crisis to concrete relief – a preliminary report on combining state abstraction and HTN planning," in *Proc. of ECP*, 2001.

[6] P. Bercher, D. Höller, G. Behnke, and S. Biundo, "More than a name? On implications of preconditions and effects of compound HTN planning tasks," in *Proc. of ECAI*, 2016.

[7] K. Erol, J. Hendler, and D. Nau, "Complexity results for HTN planning," *Annals of Math. and AI*, vol. 18, no. 1, pp. 69–93, 1996.

[8] D. McAllester and D. Rosenblitt, "Systematic nonlinear planning," in *Proc. of AAAI*, 1991.

[9] P. Bercher, D. Höller, G. Behnke, and S. Biundo, "User-centered planning – a discussion on planning in the presence of human users," in *Proc. of ISCT*, 2015.

[10] R. Byrne, "Planning meals: Problem solving on a real data-base," *Cognition*, 1977.

[11] M. Schmidt-Schauss and G. Smolka, "Attributive concept descriptions with complements," *AIJ*, vol. 48, pp. 1–26, 1991.

[12] G. Ferguson, J. Allen, and B. Miller, "TRAINS-95: towards a mixed-initiative planning assistant," in *Proc. of AIPS*, 1996.

[13] J. Bresina, A. Jónsson, P. Morris, and K. Rajan, "Activity planning for the mars exploration rovers," in *Proc. of ICAPS*, 2005.

[14] P. Bercher and D. Höller, "Interview with David E. Smith," *Künstliche Intelligenz*, vol. 30, no. 1, pp. 101–105, 2016, Special Issue on Companion Technologies.

[15] J. Fernández-Olivares, L. Castillo, Ó. García-Pérez, and F. Palao, "Bringing users and planning technology together. Experiences in SIADEX," in *Proc. of ICAPS*, 2006.

[16] K. Myers, P. Jarvis, W. Tyson, and M. Wolverton, "A mixed-initiative framework for robust plan sketching," in *Proc. of ICAPS*, 2003.

[17] G. Behnke, D. Ponomaryov, M. Schiller, P. Bercher, F. Nothdurft, B. Glimm, and S. Biundo, "Coherence across components in cognitive systems – One ontology to rule them all," in *Proc. of IJCAI*, 2015.

[18] F. Nothdurft, G. Behnke, P. Bercher, S. Biundo, and W. Minker, "The interplay of user-centered dialog systems and AI planning," in *Proc. of SIGDIAL*, 2015.

[19] G. Behnke, D. Höller, P. Bercher, and S. Biundo, "Change the plan – How hard can that be?" in *Proc. of ICAPS*, 2016.

[20] J. Williams, E. Kamal, H. Mokhtar Ashour, J. Miller, and G. Zweig, "Fast and easy language understanding for dialog systems with microsoft language understanding intelligent service (LUIS)," in *Proc. of SIGDIAL*, 2015, pp. 159–161.

[21] F. Nothdurft, P. Bercher, G. Behnke, and W. Minker, "User involvement in collaborative decision-making dialog systems," in *Dialogues with Social Robots: Enablements, Analyses, and Evaluation*, K. Jokinen and G. Wilcock, Eds. Springer, 2017.

[22] F. Nothdurft, F. Richter, and W. Minker, "Probabilistic human-computer trust handling," in *Proc. of SIGDIAL*, 2014, pp. 51–59.

[23] M. Schiller, F. Schiller, and B. Glimm, "Testing the adequacy of automated explanations of EL subsumptions," in *Proc. of DL 2017, CEUR*, vol. 1879, 2017.