Coherence Across Components in Cognitive Systems – One Ontology to Rule Them All

Gregor Behnke[*], **Denis Ponomaryov**[†], **Marvin Schiller**[*], **Pascal Bercher**[*], **Florian Nothdurft**[‡], **Birte Glimm**[*] and **Susanne Biundo**[*]
[*]Institute of Artificial Intelligence, Ulm University, Germany
[†]A.P. Ershov Institute of Informatics Systems, Novosibirsk, Russia
[‡]Institute of Communications Engineering, Ulm University, Germany

Abstract

The integration of the various specialized components of cognitive systems poses a challenge, in particular for those architectures that combine planning, inference, and human-computer interaction (HCI). An approach is presented that exploits a single source of common knowledge contained in an ontology. Based upon the knowledge contained in it, specialized domain models for the cognitive systems' components can be generated automatically. Our integration targets planning in the form of hierarchical planning, being well-suited for HCI as it mimics planning done by humans. We show how the hierarchical structures of such planning domains can be (partially) inferred from declarative background knowledge. The same ontology furnishes the structure of the interaction between the cognitive system and the user. First, explanations of plans presented to users are enhanced by ontology explanations. Second, a dialog domain is created from the ontology coherent with the planning domain. We demonstrate the application of our technique in a fitness training scenario.

1 Introduction

Our cognitive skills allow us to interact with our environment and to smoothly adapt and react to external influences. We do so by using various senses and by relying on previous experiences from other contextual situations and our ability to learn, reason, and plan future actions. Technical systems that implement or imitate the cognitive skills of humans are what we call *cognitive systems*. Such systems heavily rely on background knowledge (about the application domain) for planning actions, conducting dialogs with users, or when explaining system decisions and actions. Traditionally, each system component (e.g., the planning or dialog component) uses its own knowledge model. This distribution of knowledge makes it difficult to obtain a coherent system and often results in the redundant creation of formal knowledge in different formalisms. In this paper, we propose an integrated approach to building systems with cognitive abilities. The main distinguishing feature of the approach is that knowledge models for the system's components are automatically generated (using automated reasoning) from a centralized knowledge model in the form of an OWL ontology [W3C OWL Working Group, 2009].

We illustrate how this approach can be applied in a real-world fitness training scenario, where the ontology and automated reasoning are used to derive decomposition methods for the planning domain in the context of Hierarchical Task Network (HTN) planning. Hence, any standard HTN planning system can be used with the generated domain. We then sketch how the same ontology supports the task of explaining system behavior. To explain plan steps and decompositions that are hard to explain solely based on a standard planning domain, one can verbalize the ontology reasoning that led to them. Further, the same ontology contributes to the creation of a dialog domain. Using this integrated approach, a coherent human-machine interface can be realized, while avoiding the creation of redundant knowledge in different components. In the context of HCI and dialog modeling, it was shown that interactive systems benefit from increased users' trust if they provide additional explanations for their actions [Lim *et al.*, 2009; Nothdurft *et al.*, 2014]. Bercher *et al.* [2014] show how plan explanations can be applied successfully in a real-world scenario.

The paper is organized as follows: First, we introduce some preliminaries about planning and ontologies (Sec. 2). In Section 3, we present the foundations of the proposed approach and cover how planning domains can be encoded in an ontology such that new decomposition methods can automatically be inferred by an ontology reasoner. This chapter also introduces our use-case – a fitness training scenario – and serves to illustrate how our technique can be applied to it. We then describe how plan explanations can be enriched with knowledge from the ontology (Sec. 4). Thereafter we introduce the dialog component (Sec. 5) used in the context of our scenario and outline how such a component can be generated in general. We discuss related approaches in Section 6 and conclude in Section 7.

2 Preliminaries

We now introduce some relevant preliminaries.

2.1 HTN Planning

HTN planning [Erol *et al.*, 1994] has been successfully applied to many real-world problems [Nau *et al.*, 2005]. We

deem it appropriate for a cognitive system that heavily interacts with a user, since the top-down way in which initially abstract tasks are step-wise refined into more concrete courses of action is similar to human problem solving. In HTN planning, the problem to solve is given by means of an initial plan containing a description of the initial state (a description of the world properties that are true prior to the execution of a plan and the interaction with the system), a goal description (stating the desired world properties after plan execution), and a partially ordered set of tasks (the initial plan). These tasks may be primitive, meaning that they can be directly executed by the user, or they may be abstract (also referred to as complex or compound in the literature). Primitive tasks are given in terms of their preconditions and effects, specifying the conditions under which they are applicable in a state and how they change it if applied. Preconditions and effects are specified in terms of lists of literals of a function-free first order logic. Abstract tasks represent high-level activities that have to be refined into more tangible tasks, being primitive or abstract. Refining abstract tasks is achieved by applying a so-called (decomposition) method specifying into which plan the respective task can be decomposed. More precisely, a method m is denoted as $\mathcal{A} \mapsto_{\prec} \mathcal{B}_1, \ldots, \mathcal{B}_n$ and it specifies that the abstract task $\mathcal{A}$ may be decomposed into the plan containing the subtasks $\mathcal{B}_1$ to $\mathcal{B}_n$ that are ordered w.r.t. the partial order $\prec$. We omit the subscript $\prec$ if no order is defined on the subtasks. The application of m to a plan containing $\mathcal{A}$ refines it into a plan in which $\mathcal{A}$ is "replaced" by the subtasks of m with the given order $\prec$. Any ordering that was imposed on $\mathcal{A}$ is inherited by all its subtasks $\mathcal{B}_i$. A solution of the problem is given by any plan that fulfills the goals. That is, a solution must be obtained from the initial plan, it has to be primitive (as only primitive tasks are regarded executable by the user), and any linearization of its tasks that is compatible with the given order must be executable and transforms the initial state into a state satisfying the goal description.

2.2 Ontologies

Ontologies based on Description Logics (DLs) are often used to represent knowledge about *concepts* and relations (*roles*) between them in a subject domain. DLs also underpin the W3C standard OWL (Web Ontology Language). In this paper, we use the Manchester OWL syntax [Horridge *et al.*, 2006] and consider the fragment of OWL corresponding to the DL $\mathcal{ALC}$ [Schmidt-Schauss and Smolka, 1991]. It allows for building complex concepts from primitive ones, e.g., by using `and`/`or` connectives and quantifier-like `some`/`only` constructs, which form concepts by specifying a relationship to other concepts via a role. Since DLs are fragments of first order logic, their semantics is given model-theoretically, with concepts interpreted as subsets of a domain and roles as binary relations. For example, for a concept *Course* and a role *teaches*, the concept (*teaches* `some` *Course*) is interpreted as the set of all domain individuals that are related by *teaches* to *some* instance of *Course*. The interpretation of (*teaches* `some` *Course*) `and` (*teaches* `only` *Course*) is the set of individuals which meet the additional requirement (due to the conjunction `and`) that they are *only* related via *teaches* to instances of *Course*. The two reserved primitive concepts `Thing` and `Nothing` are interpreted as the universe and empty set, respectively.

An *ontology* or *knowledge base* $\mathcal{O}$ is a finite set of concept axioms of the form `Class:` C `SubClassOf:` D, stating that every instance of the concept C is also an instance of the concept D, definitions of the form `Class:` C `EquivalentTo:` D, stating that D subsumes C and vice versa and disjointness axioms of the form `Class:` C `DisjointWith:` D, stating that no instance of C can be an instance of D and vice versa. An interpretation that satisfies all axioms of $\mathcal{O}$ is called a *model* of $\mathcal{O}$. A basic reasoning task is to decide for a given ontology $\mathcal{O}$ and concepts C, D, whether C is *subsumed* by D wrt $\mathcal{O}$, i.e., whether the concept inclusion is logically entailed by $\mathcal{O}$. Another important task is to check whether a concept is *satisfiable* wrt $\mathcal{O}$, i.e., whether it can have instances. If it is clear from the context, we omit the reference to $\mathcal{O}$.

3 Integrating Declarative Knowledge

In this section we show how the core knowledge of a cognitive system can be encoded in an ontology and how the central problem-solving component, in our case a planner, can retrieve this knowledge. The two subsequent sections describe how the same ontology can be used to ensure coherent knowledge in two other components of a cognitive system, i.e., the explanation and the dialog components. We start by describing how a (pre-existing) planning domain can be integrated (some parts only syntactically) into an ontology of the application domain. Thereafter, specific domain knowledge can be drawn from this ontology, i.e., it serves as the sole source of application knowledge ensuring coherence. Extracting a planning domain from an ontology is straightforward.

In addition to the obvious advantages for cognitive systems, encoding the planning domain into an ontology has additional benefits. Most importantly, it enables inferring new decomposition methods for hierarchical planning domains. Usually, every decomposition method has to be specified by a domain expert, making it a slow and complicated process. Generating methods automatically eases the modeling of hierarchical domains and enables on-the-fly changes to domains without the need to consult a modeler. Only a very limited number of approaches to do so have been published so far, see Section 6 for further details.

3.1 Exemplary Use-Case

In order to exemplify potential uses and overall benefits of our approach, as well as to provide illustrative examples we interleave the formal descriptions in this section with the description of a use-case of the technique. We have chosen a fitness-training scenario, in which a user wants to create a training plan with the help of a planning system in order to pursue some fitness objective, e.g., to increase his overall strength or stamina. A plan in this scenario defines a training schedule, comprising training and rest days, as well as the exercises and their duration which are necessary to achieve a given goal. Pulido *et al.* [2014] considered a similar scenario, where physiotherapy exercises are arranged by a planner to help patients recover from upper-limb injuries.

In this scenario, we distinguish four kinds of actions: exercises, workouts, workout templates, and trainings. *Exer-*

cises are concrete physical activities, e.g., skip rope jumping. *Workouts* are small, predefined partially ordered sets of exercises, which usually have been created by a fitness trainer and should be performed in a single training session. *Workout templates* are more abstract descriptions of workouts, intended to group "similar" workouts. Finally, *trainings* describe abstract objectives like strength or lower body training.

3.2 Compiling Planning Knowledge into Ontologies

We start by outlining how a hierarchical planning domain can be encoded in an ontology such that its contents are described declaratively (and become amenable to logical reasoning). Conceptually, we augment an already existing ontology. This ontology is presumed to contain further background knowledge about the application domain. In our use-case the ontology already contains parts of the taxonomy of the NCICB corpus [NCICB, 2015], which describes muscles, joints and bones of the human body and their relations.

The link between the planning model and corresponding information in the ontology is maintained by a common vocabulary – for every planning task $\mathcal{T}$ a corresponding concept T is added to the ontology. Concepts representing primitive tasks need to specify preconditions and effects, which can be modeled in two ways. Where a shallow granularity of modeling is deemed sufficient, predicates in the planning domain are represented as string values of four pre-defined OWL data properties (also known as concrete roles); *adds*, *deletes*, *needs* and *hindered-by*. Any parameters of the predicates (and possible restrictions) are represented in the ontology only in the form of annotations. A more straightforward representation of the first-order axiomatization of tasks in general is prevented by the limitation of DLs to the tree model property [Vardi, 1997] stating that non-tree structures cannot fully be axiomatized, which would be required. Thus, to allow for a second, more fine-grained, but domain-dependent, way of modeling, (complex) concepts in the ontology can be defined to correspond (using a dedicated mapping) to specific sets of preconditions and effects in the planning domain. For example, one can specify that a concept description such as (*trains* `some` *GastrocnemiusMuscle*) (anything that trains the gastrocnemius muscle) maps to a precondition *warmedup(GastrocnemiusMuscle)* and an effect *trained(GastrocnemiusMuscle)*. As an example of a definition of a primitive exercise, consider the skip rope jumping exercise, which is defined in terms of NCICB concepts (marked by underlining) and which serves to specify which muscles are engaged and trained by it.

`Class:` *SkipRopeJumping*
`SubClassOf:` *trains* `some` *GastrocnemiusMuscle*
`and` *engages* `some` *QuadricepsFemorisMuscle*
`and` *engages* `some` *Hamstring*

This definition implies that the *SkipRopeJumping* action has as a precondition that the gastrocnemius muscle must be warmed up. The effect of this action is that this muscle is trained and the two other muscles are warmed up. In our use-case domain, we utilize these preconditions and effects to model certain training rules (e.g. muscles must be warmed up before being used in exercises, more intense exercises before lighter ones, …).

The most important part of a hierarchical planning domain to be represented in the ontology is the hierarchy itself. Our approach is guided by the principle that individuals, i.e., the objects belonging to a concept, represent plans. As such, concepts represent a set of plans, i.e., all plans that can be obtained by decomposing the task they represent. Following this intuition, concept subsumption between two concepts *A* and *B* can be interpreted as: any plan obtainable from B is also obtainable from A. Thus, simple methods of the form $\mathcal{A} \mapsto \mathcal{B}$, so-called unit methods, are interpreted as $\mathcal{B}$ being a specialization of $\mathcal{A}$, since $\mathcal{A}$ can be achieved by "executing" $\mathcal{B}$. It is represented as `Class:` B `SubClassOf:` A. In general a method with multiple subtasks specifies that the abstract task $\mathcal{A}$ can be decomposed into several other tasks $\mathcal{B}_1, \ldots, \mathcal{B}_n$. The relation "can be decomposed into" is represented in the ontology by using a role *includes*. When representing such decomposition methods in an ontology, however, one needs to take into account that a decomposition specifies not only what subtasks are needed to achieve an abstract task, but that these are also *sufficient*. By contrast, ontologies are built on the open world assumption – representing a task decomposition simply by stating that an abstract task can be decomposed into some particular set of tasks is insufficient. One also needs to explicitly state that only these tasks are to be included in the decomposition. Suppose, for instance, that an abstract task for a workout of the lower body, which contains skip rope jumping and stationary bike exercises, can be represented by

`Class:` *LowerBodyWorkout*
`EquivalentTo:` *includes* `some` *SkipRopeJumping* `and`
includes `some` *StationaryBikeExercise*

Using DL reasoning one could infer that a workout containing skip rope jumping, stationary bike exercises and also push ups would be a sub-concept of *LowerBodyWorkout*. This, in turn, can be interpreted as the assertion that said workout could be used (i.e. *LowerBodyWorkout* could be decomposed into it) as a lower body workout which should clearly not be the case.

Intuitively, we want a collection C_1 (representing tasks in a plan) to be subsumed by a collection C_2 iff for any task concept (requirement) from C_2, there is a task concept in C_1, which achieves it (i.e. C_1 is subsumed by C_2), and there are only those task concepts in C_1 that meet some requirement from C_2. This calls for using the `onlysome` construct (see e.g. Horridge *et al.* [2006]), which is provided by the Manchester OWL syntax as a macro. The macro is of the form r `onlysome` $[C_1, \ldots, C_n]$, which expands to r `some` C_1 `and` … `and` r `some` C_n `and` r `only` $(C_1$ `or` … `or` $C_n)$. Using `onlysome` we can describe a method decomposing a task $\mathcal{A}$ into a plan containing the tasks $\mathcal{B}_1, \ldots, \mathcal{B}_n$ by the axiom `Class:` A `EquivalentTo:` *includes* `onlysome` $[B_1, \ldots, B_n]$. If we apply this scheme to the *LowerBodyWorkout*, we obtain the following axiom, which (as proven below) correctly

reflects our intuition.

Class: *LowerBodyWorkout*
EquivalentTo: *includes* onlysome [*SkipRopeJumping*, *StationaryBikeExercise*]

This axiom is semantically equivalent to

Class: *LowerBodyWorkout*
EquivalentTo: *includes* some *SkipRopeJumping* and *includes* some *StationaryBikeExercise* and *includes* only (*SkipRopeJumping* or *StationaryBikeExercise*)

Furthermore, the onlysome construct allows a domain expert to specify trainings in an abstract way without the need to include every potentially possible decomposition into workouts in the planning domain. This abstract modeling enables inferring new decomposition methods for trainings into workouts, which achieve the encoded training objective. The following axiom defines lower body training as anything that contains at least one and only exercises targeting muscles in the lower body.

Class: *LowerBodyTraining*
EquivalentTo: *includes* onlysome [*trains* some (*partOf* some *LowerBody*)]

Here the intended subsumption between *LowerBodyWorkout* and *LowerBodyTraining* holds since both *SkipRopeJumping* and *StationaryBikeExercise* engage only parts of the lower body.

This approach of translating decomposition methods into ontology axioms hinges on whether subsumption in our representation correctly reflects our intuitions about whether collections of tasks fulfill the "requirements" imposed by another (possibly more abstract) collection of tasks. We show that in order for this property to hold, it is required that the role *includes* is *independent* of all concepts occurring in onlysome constructs. Independence holds if a role r has no semantic relationship with the concepts $C_1, \ldots, C_n$ in the onlysome construct as captured by the following definition.

Definition 1. *Let $\mathcal{O}$ be an ontology, r a role, and $C_1, \ldots, C_n$ concepts. We call r* independent of $C_1, \ldots, C_n$ wrt $\mathcal{O}$ *if, for any model $\mathcal{I}$ of $\mathcal{O}$ and any binary relation $[s]$ on the domain of $\mathcal{I}$, there is a model $\mathcal{J}$ of $\mathcal{O}$ with the same domain such that r is interpreted as $[s]$ in $\mathcal{J}$ and the interpretation of C_i, $1 \leq i \leq n$, in $\mathcal{I}$ and $\mathcal{J}$ coincides.*

Theorem 1. *Let $\mathcal{O}$ be an ontology, $C_1, \ldots, C_m$ satisfiable concepts, $D_1, \ldots, D_n$ concepts, and r a role independent of $C_1, \ldots, C_m, D_1, \ldots, D_n$. Then r onlysome $[D_1, \ldots, D_n]$ subsumes r onlysome $[C_1, \ldots, C_m]$ if and only if*
(1) $\forall i, 1 \leq i \leq m, \exists j, 1 \leq j \leq n$, s.t. D_j subsumes C_i and
(2) $\forall j, 1 \leq j \leq n, \exists i, 1 \leq i \leq m$, s.t. D_j subsumes C_i.

Proof Sketch. The if direction can be shown using monotonicity and above Conditions *(1)* and *(2)*. Using contraposition, we can show the only-if direction by constructing a model of the ontology $\mathcal{O}$ in which the subsumption does not hold. Since each C_i is satisfiable, there are models with some instance c_i of C_i. Using the negation of Condition *(1)*, there is further a model with an instance x of some C_i that is not an instance of any D_j. We now build a new model as the disjoint union of these models [Baader *et al.*, 2003, p. 195] and take an arbitrary element d in the constructed model. Using independence of r, we obtain a model in which r contains $\langle d, x \rangle$ and the tuples $\langle d, c_i \rangle$. We obtain the desired contradiction since d is an instance of r onlysome $[C_1, \ldots, C_m]$, but d is not an instance of r only $(D_1$ or ... or $D_m)$ (due to $\langle d, x \rangle$) and, hence, r onlysome $[D_1, \ldots, D_n]$. We can proceed similarly for the case of Condition *(2)* not holding. □

So far, we have only dealt with the tasks contained in decomposition methods, but not with their order. In the general case, ordering constraints represent partial orders and impose dependencies in the form of a directed acyclic graph between tasks. OWL is not suited to represent such partial orders, since its expressivity is limited by the previously mentioned tree-model property. In order to include ordering constraints nevertheless, we propose a syntactic encoding for this information that is opaque to DL reasoners and has no influence on the semantics. A task $\mathcal{A}$ occurring after a task $\mathcal{B}$ in a plan is expressed by replacing the concept *A* in onlysome expressions by *A* or (Nothing and after some *B*). Note that the latter disjunct is trivially unsatisfiable and, consequently, the concept is semantically equivalent to just *A*.

3.3 Generating Planning Domains

This section describes how the ontology is utilized to infer new decomposition methods for an existing planning domain. The decision for representing the whole planning domain in the ontology enables using off-the-shelf DL reasoners [Gonçalves *et al.*, 2013, Section 3]. In keeping with the encoding introduced in the previous section, we view subsumption relations between tasks inferred by a reasoner as new decomposition methods. Suppose there are two task concepts *C* and *D*, such that *C* is subsumed by *D* and no other task concept exists that is subsumed by *D* and subsumes *C*. Then, a decomposition method $\mathcal{D} \mapsto \mathcal{C}$ is created. This simple scheme alone does not suffice, as it only creates methods with a single subtask. As a next step, we can interpret onlysome-definitions provided by the ontology modeler as decompositions, too. This may add new tasks to the planning domain, as the onlysome construct may contain an arbitrary OWL expression *E* without a corresponding planning task. If so, the task $\mathcal{E}$ is added to the planning domain and treated as a named concept in the ontology.

We are also interested in knowing whether an abstract task $\mathcal{A}$ can be achieved by *combining* some other tasks $\mathcal{B}_1, \ldots, \mathcal{B}_n$. If so, a new decomposition method for $\mathcal{A}$ into $\mathcal{B}_1, \ldots, \mathcal{B}_n$ can be created. This capability has proven useful in our use-case domain to be able to combine two workouts to achieve some common goal. Consider the following definition of a full body

training:

Class: *FullBodyTraining*
EquivalentTo: *includes* onlysome
[*trains* some (*partOf* some *LowerBody*),
trains some (*partOf* some *UpperBody*)]

Further suppose that the model contains only workouts which exclusively train parts of the upper and the lower body, since workouts often target a specific group of muscles or a particular part of the musculoskeletal system. Here it would be necessary to combine, e.g., the previously defined *LowerBodyWorkout* with a workout for the upper body, say *UpperBodyWorkout*.

Again, we want to connect this to subsumption of some concepts. As a first naive idea, one could model the combination of concepts as their conjunction, and check whether these conjunctions are subsumed by other concepts. However, in the case of collections of task concepts (using onlysome) this simple idea does not work – the conjunction of two onlysome expressions in general is not equivalent to an onlysome expression containing the same elements. For example, consider the expressions $E_1 =$ *includes* onlysome $[A, B]$, $E_2 =$ *includes* onlysome $[A']$ and $E_3 =$ *includes* onlysome $[B']$, where A' and B' are sub-concepts of A and B, respectively. Then neither E_2 and E_3 nor *includes* onlysome $[E_2, E_3]$ is subsumed by E_1 as desired. In fact, OWL does not provide a connective that can directly be used to combine E_2 and E_3, but the concept $E_4 =$ *includes* onlysome $[A', B']$ that uses the sub-concepts of E_2 and E_3 is subsumed by E_1 as intended. Due to the lack of a suitable OWL operator, we next define a new, syntactic join operator to combine concepts.

Definition 2. *Given two concepts* $E_1 = r$ onlysome $[A_1, \ldots, A_n]$ *and* $E_2 = r$ onlysome $[B_1, \ldots, B_m]$*, we define the* join *of* E_1 *and* E_2 *wrt* r*, written* E_1 r-join E_2*, as* r onlysome $[A_1, \ldots, A_n, B_1, \ldots, B_m]$.

Using this definition we can combine the workouts *LowerBodyWorkout* and *UpperBodyWorkout* and infer that they are subsumed by the *FullBodyTraining* concept, which is a useful decomposition method.

To sum up, any conjunction of task concepts, any onlysome connection of task concepts, and any join of task concepts can be considered as candidate subtask sets for new decomposition methods. If such an expression is subsumed by a task concept in the ontology, the corresponding decomposition method is added to the planning domain, except if a combination of a subset of the concepts included in the candidate is already subsumed. Considering all possible combinations presents us with a problem if the domain in question has a realistic size, since there are exponentially many. To circumvent this problem, we propose a pragmatic approach. First, the maximal size of connections of concepts can be restricted by some k, which reduces the number of concepts to be added considerably. For our application example we only used binary ($k = 2$) combinations, which already allowed for inferring a considerable number of tasks and methods. Second, most real-world domains (including our application example) have restrictions on which concepts may be combined. Thus, the set of concepts can be partitioned (e.g. via OWL annotations) and only concepts of the same partition are combined, leading to a further reduction of the number of candidates.

3.4 Evaluation and Discussion

In our case-study scenario the initial, non-extended planning domain contains 310 different tasks and only a few methods. The described ontology contains 1230 concepts (613 imported from the NCICB corpus) and 2903 axioms (of which 664 are from NCICB). This includes 310 concepts for integrating the planning domain into the ontology. Further, the ontology contains 9 different training objectives and 24 workout templates.

The planning domain, expanded with new decompositions inferred from the ontology, contains 471 tasks and 967 methods. Our implemented system employs the OWL reasoner FaCT++ [Tsarkov and Horrocks, 2006]. On an up-to-date laptop computer (Intel® Core™ i5-4300U) it takes 3.6 seconds to compute the whole extended planning domain.

Of the newly generated methods, 203 are created based upon workouts subsumed by workout templates and 3 methods have been created by combinations of concepts. Further, 59 decomposition methods for training objectives into workout templates have been found of which 24 are combinations of concepts. We would like to point out that every decomposition linking workouts and trainings in this scenario is inferred by the reasoner.

4 Ontology and Plan Explanations

The explanation facility combines techniques from plan explanation (specifically, an approach for explaining hybrid plans [Seegebarth *et al.*, 2012]) and an approach for explaining ontological inferences [Schiller and Glimm, 2013]. The first approach is best suited to explain the dependencies of tasks in the generated plan. The second makes the underlying principles and further background knowledge relevant to the plan explicit, in particular, how decomposition methods are generated.

In more detail, the user may be interested in knowing why a task is part of a plan, for example "Why do I have to do a runners' calf stretch?" when a plan contains such a task. This question is taken to address the causal and hierarchical structure of the plan, and the corresponding explanation is directly generated from the causal dependencies contained in and the decompositions applied to the plan (as shown for the example in Figure 1). Such an explanation is an ordered list of arguments, where a causal dependency is expressed as "Task A was necessary, as it establishes l needed by B" and a decomposition as "Task A was necessary, since it must be executed to achieve B". In the running example, the following explanation is generated:

```
The runners' calf stretch is necessary as
it ensures that the gastrocnemius muscle is
warmed up, which is needed by the skip rope
jumping. The skip rope jumping is necessary,
since it is part of the lower body workout.
```

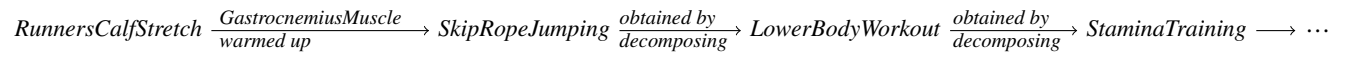

Figure 1: Structure of a formal plan for the running example, as used in the explanation

`The lower body workout is necessary, since it is part of the stamina training.`

Such explanations treat decomposition methods as facts, but they are not justified further. While those generated from causal dependencies may be plausible to a human, those generated from decompositions might not. For instance, in the running example, the user may further ask "Why is the lower body workout a stamina training?". We use the second explanation mechanism to further justify decompositions, which are represented by subsumption relationships of the form `Class:` A `SubClassOf:` C and which are logically implied by the ontology. Here the aim is to present a stepwise explanation making relevant background knowledge explicit. Therefore, a derivation tree for the subsumption is constructed from the relevant domain axioms using a consequence-based reasoning mechanism. To generate verbal output (with the goal of imitating natural language), the nodes in this tree are first ordered in a linear fashion. Each inference rule specifies a template according to which its premises and conclusions are output or simply omitted. Formulas that occur as part of the premises or as the conclusion are converted into phrases by applying patterns similar to those used by Nguyen [2013], together with some mechanisms for aggregation. In our running example, the following justification is provided to the user:

```
According to its definition, the lower
body workout includes skip rope jumping and
stationary bike exercise. Furthermore, since
skip rope jumping is an aerobic exercise, it
follows that the lower body workout includes
an aerobic exercise. Given that something
that includes an aerobic exercise has stamina
as an intended health outcome, the lower body
workout has stamina as an intended health
outcome. Thus, the lower body workout is a
stamina training according to the definition
of stamina training.
```

This explanation can be considered more verbose than needed, since it encompasses all the relevant information that formally proves the relationship under question. Future work should address adjustments to the level of detail based on user modeling and pragmatics. Our approach shares its main ideas with related work by Nguyen [2013], who also uses stepwise derivations and template-based verbalization to generate explanations for inferences in ontologies. Both Nguyen and the present work rely on further related work by Horridge [2011] to pinpoint those axioms that are relevant for the explanations that are to be presented. The present work contrasts with a number of other approaches that mainly focus on how ontology axioms are suitably verbalized in natural language (so-called ontology verbalization, cf. [Androutsopoulos *et al.*, 2013]), but that leave inference aside.

Since both the plan-based and the ontology-based explanation mechanisms we use share the same integrated representation, one can present enhanced and still consistent explanations to the user.

5 Integrating the User

In order to integrate the user into the planning process and to communicate the generated solution, a dialog management component is needed to control the flow and the structure of the interaction. In order to communicate a solution, all planned tasks have to be represented in the dialog domain, while integrating the user requires the ongoing presentation of planning decisions. This includes, most notably, the choice of a decomposition method if an abstract task is to be refined. The use of shared knowledge considerably facilitates coherency of the interaction. Although the planning knowledge stored in the ontology alone is not sufficient for the generation of the dialog domain, it contributes to its structure and enables a unisono view on the domain, eliminating inconsistency and translation problems.

The integrated planning knowledge, used to infer new decompositions for existing planning domains, can be used to create a basic dialog structure as well. Analogous to Section 3, a dialog $\mathcal{A}$ can be decomposed into a sequence of subdialogs containing the dialogs $\mathcal{B}_1, \ldots, \mathcal{B}_n$ by an axiom `Class:` A `EquivalentTo:` *includes* `onlysome` $[B_1, \ldots, B_n]$. For example, in our application scenario a strength training can be conducted using a set of workouts $\mathcal{A}_1, \ldots, \mathcal{A}_n$, each of which consists of a set of exercises $\mathcal{B}_1, \ldots, \mathcal{B}_n$. This way a dialog hierarchy can be created, using the topmost elements as entry points for the dialog between user and machine. However, in addition to the knowledge used to generate plan steps, additional resources are required for communicating these steps to the user. Such texts, pictures or videos can easily be referenced from an ontology. Using this information, dialogs suitable for a well-understandable human-computer interaction can be created and presented to the user.

One key aspect of state-of-the-art dialog systems is the ability to individualize the ongoing dialog according to the user's needs, requirements, preferences or history of interaction. Coupling the generation of the dialog domain to the ontology enables us to accomplish these requirements using ontological reasoning and explanation in various ways. The dialogs can be pruned using ontological reasoning according to the user's needs (e.g. show only exercises which do not require gym access), to the user's requirements (e.g. show only beginner exercises), or adapted to the user's dialog history (e.g. preselect exercises which were used the last time) and preferences (e.g. present only exercises with dumbbells). Integrating pro-active as well as requested explanations into the interaction is an important part of imparting used domain knowledge and clarifying system behavior. Using a coherent knowledge source to create dialog and planning domain enables us to use predefined declarative explanations [Nothdurft

et al., 2014] together with the plan explanations described in Section 4, without dealing with inconsistency issues.

6 Related Work

Past research on coupling ontological reasoning and planning mainly focused on increasing the planners efficiency or the languages expressivity. A survey of Gil [2005] describes approaches joining classical planning and ontologies and lists several planners that use ontological reasoning to speed-up plan generation. Hartanto and Hertzberg [2008] use an ontology to prune a given HTN model by deleting non-reachable constants. It is not possible to infer additional content for the domain within their paradigm. Several approaches use ontologies to enrich the structure of the planning domain. Ontologies provide hierarchies of tasks and plans and are used to represent states with the open world assumption [Sánchez-Ruiz *et al.*, 2009; Sirin, 2006]. For a survey, we refer to Sirin [2006, Chapter 8].

Sirin [2006] proposes the HTN-DL formalism, an integration of HTN and description logics to address Web Service composition problems. An HTN-DL planning domain is related to an ontology representing tasks and decomposition methods as concepts and individuals, respectively, which are augmented with an additional structure to encode parameter variables, preconditions, and effects. In that Sirin's notion of methods is different from standard HTN, as they do not specify which abstract task they decompose but merely contain a partially ordered list of actions. Instead, his methods have preconditions and effects, like ordinary actions, which are not necessarily related to the contents of the plan. This potential inconsistency could be circumvented by using a legality criterion for decompositions (cf. [Biundo and Schattenberg, 2001]). Although Sirin's and our approach are similar in the idea of using an ontology and DL reasoning to generate planning domains, there are conceptual differences. Most notably, Sirin assumes that all decomposition methods are given in advance by a domain modeler, while our approach can infer completely new decomposition methods. In HTN-DL, reasoning is applied to match a decomposition method to a task to which it may be applied. This matching is based solely on the preconditions and effects of tasks and methods and not on the tasks contained in the method. It does not allow a deep reasoning about these tasks and their decompositions or other properties, which is possible with our approach. So far, there has been only little work on inferring decomposition methods automatically, one example being the work of Knoblock [1994].

7 Conclusion

We have demonstrated how an ontology, used to generate planing and dialog domains, facilitates coherence in a cognitive system and at the same time advances its explanation capabilities. An application scenario was described and the benefits of our approach outlined.

In future work we will integrate more advanced planning techniques into the system. Most notably, this includes mixed-initiative planning. Also we would like to tackle the task of integrating order directly into the reasoning process, despite the problems discussed. Our work raises the issue of verbosity and pragmatics for plan explanations to be addressed in further studies.

Acknowledgments

We want to thank the reviewers for their help to significantly improve the paper. This work was done within the Transregional Collaborative Research Centre SFB/TRR 62 "A Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

[Androutsopoulos *et al.*, 2013] Ion Androutsopoulos, Gerasimos Lampouras, and Dimitrios Galanis. Generating natural language descriptions from OWL ontologies: The NaturalOWL system. *JAIR*, 48:671–715, 2013.

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Bercher *et al.*, 2014] Pascal Bercher, Susanne Biundo, Thomas Geier, Thilo Hoernle, Florian Nothdurft, Felix Richter, and Bernd Schattenberg. Plan, repair, execute, explain - how planning helps to assemble your home theater. In *Proc. of the 24th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 386–394. AAAI Press, 2014.

[Biundo and Schattenberg, 2001] Susanne Biundo and Bernd Schattenberg. From abstract crisis to concrete relief (a preliminary report on combining state abstraction and HTN planning). In *Proc. of the 6th Europ. Conf. on Planning (ECP)*, pages 157–168, 2001.

[Erol *et al.*, 1994] Kutluhan Erol, James A. Hendler, and Dana S. Nau. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proc. of the Int. Conf. on AI Planning & Scheduling (AIPS)*, pages 249–254, 1994.

[Gil, 2005] Yolanda Gil. Description logics and planning. *AI Magazine*, 26(2):73–84, 2005.

[Gonçalves *et al.*, 2013] Rafael Gonçalves, Samantha Bail, Ernesto Jiménez-Ruiz, Nicolas Matentzoglu, Bijan Parsia, Birte Glimm, and Yevgeny Kazakov. OWL reasoner evaluation (ORE) workshop 2013 results: Short report. In *Proc. of the Int. Workshop on OWL Reasoner Evaluation*, volume 1015, pages 1–18. CEUR, 2013.

[Hartanto and Hertzberg, 2008] Ronny Hartanto and Joachim Hertzberg. Fusing DL reasoning with HTN planning. In *KI 2008: Advances in AI*, volume 5243 of *LNCS*, pages 62–69. Springer, 2008.

[Horridge *et al.*, 2006] Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, Robert Stevens, and Hai Wang. The Manchester OWL syntax. In *Proc. of the Workshop on OWL Experiences and Directions*, Athens, GA, USA, 2006.

[Horridge, 2011] Matthew Horridge. *Justification Based Explanations in Ontologies*. PhD thesis, University of Manchester, Manchester, UK, 2011.

[Knoblock, 1994] Craig A. Knoblock. Automatically generating abstractions for planning. *Artificial intelligence*, 68(2):243–302, 1994.

[Lim *et al.*, 2009] Brian Y. Lim, Anind K. Dey, and Daniel Avrahami. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proc. of the SIGCHI Conf. on Human Factors in Comp. Systems*, pages 2119–2128, 2009.

[Nau *et al.*, 2005] Dana S. Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, Héctor Muñoz-Avila, J. William Murdock, Dan Wu, and Fusun Yaman. Applications of SHOP and SHOP2. *IEEE Intelligent Systems*, 20(2):34–41, 2005.

[NCICB, 2015] NCI Center for Bioinformatics NCICB, 2015. `http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl` (accessed February 9, 2015).

[Nguyen, 2013] Tu Anh T. Nguyen. *Generating Natural Language Explanations For Entailments In Ontologies*. PhD thesis, The Open University, Milton Keynes, UK, 2013.

[Nothdurft *et al.*, 2014] Florian Nothdurft, Felix Richter, and Wolfgang Minker. Probabilistic human-computer trust handling. In *Proc. of the Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 51–59. Association for Computational Linguistics, 2014.

[Pulido *et al.*, 2014] José C. Pulido, José C. González, Arturo González-Ferrer, Javier García, Fernando Fernández, Antonio Bandera, Pablo Bustos, and Cristina Suárez. Goal-directed generation of exercise sets for upper-limb rehabilitation. In *Proc. of the Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 2014.

[Sánchez-Ruiz *et al.*, 2009] Antonio A. Sánchez-Ruiz, Pedro A. González-Calero, and Belén Díaz-Agudo. Abstraction in knowledge-rich models for case-based planning. In *Case-Based Reasoning Research and Development*, volume 5650 of *LNCS*, pages 313–327. Springer, 2009.

[Schiller and Glimm, 2013] Marvin Schiller and Birte Glimm. Towards explicative inference for OWL. In *Proc. of the Int. Description Logic Workshop*, volume 1014, pages 930–941. CEUR, 2013.

[Schmidt-Schauss and Smolka, 1991] Manfred Schmidt-Schauss and Gert Smolka. Attributive concept descriptions with complements. *AIJ*, 48:1–26, 1991.

[Seegebarth *et al.*, 2012] Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo. Making hybrid plans more clear to human users – a formal approach for generating sound explanations. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 225–233. AAAI Press, 2012.

[Sirin, 2006] Evren Sirin. *Combining Description Logic Reasoning with AI Planning for Composition of Web Services*. PhD thesis, University of Maryland at College Park, 2006.

[Tsarkov and Horrocks, 2006] Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner: System description. In *Proc. of the Third Int. Joint Conf. on Automated Reasoning (IJCAR)*, pages 292–297, Berlin, Heidelberg, 2006. Springer-Verlag.

[Vardi, 1997] Moshe Y. Vardi. Why is modal logic so robustly decidable? In *Descriptive Complexity and Finite Models*, volume 31, pages 149–184. American Mathematical Society, 1997.

[W3C OWL Working Group, 2009] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. 2009. Available at `http://www.w3.org/TR/owl2-overview/`.