An Ontological Formalization of the Planning Task

Dnyanesh RAJPATHAK and Enrico MOTTA
Knowledge Media Institute, The Open University,
Walton Hall, Milton Keynes, MK7 6AA, UK.
{d.g.rajpathak, e.motta}@open.ac.uk

Abstract. In this paper we propose a generic task ontology, which formalizes the space of planning problems. Although planning is one of the oldest researched areas in Artificial Intelligence and attempts have been made in the past at developing task ontologies for planning, these formalizations suffer from serious limitations: they do not exhibit the required level of formalization and precision and they usually fail to include some of the key concepts required for specifying planning problems. In contrast with earlier proposals, our task ontology formalizes the nature of the planning task independently of any planning paradigm, specific domains, or applications and provides a fine-grained, precise and comprehensive characterization of the space of planning problems. Finally, in addition to producing a formal specification we have also operationalized the ontology into a set of executable definitions, which provide a concrete reusable resource for knowledge acquisition and system development in planning applications.

Keywords: Planning, Ontologies, Task Ontologies, Reusable Knowledge Components, Knowledge Engineering, Knowledge Modelling.

1 Introduction

In this paper we propose a generic task ontology, which formalizes the space of planning problems. Planning is one of the oldest researched areas in Artificial Intelligence (AI). Indeed one of the earliest systems in AI, the General Problem Solver [16] was concerned with simulating human thinking processes as a planning activity. Over the years, various planning paradigms have emerged, such as *Classical Planning* [9], *Decision Theoretic Planning* (DTP) [3], *Hierarchical Task Networks* (HTN) [5]. In classical planning, the main aim of the planning task is to attain a goal-state, which is usually specified in terms of a number of desired properties of the world. Planning applications in the classical paradigm are characterized by a set of initial conditions, and a number of planning operators are then applied to transform an initial-state into a goal-state through a sequence of steps. These planning operators are typically referred to as STRIPS-operators [9]. A STRIPS-operator is a parameterized template for a set of actions and contains a set of *preconditions* that must be satisfied before an action is executed and a set of *effects* that an action can have on the world. However the classical planning approach fails to handle the dynamic nature of real-world applications and provides limited expressiveness to create efficient domain representations [21]. The DTP approach can be seen as a specialized extension of classical planning, which models the planning problem by allowing actions having uncertain effects. In this approach the planning agent may also have incomplete information about the world. This type of representation is particularly suitable to tackle problems in which the goal state is only ap-

proximately defined. At the execution level, the primary aim of the DTP paradigm is to construct a plan that has a high expected utility factor[1] rather than simply generating a plan that can achieve a goal. The planning paradigm in general and DTP in particular can be seen as a technique for solving sequential decision problems, such as decision analysis, economics, and operations research. DTP approaches use Markov Decision Processes (MDP) [7] to reason in face of uncertainty. However, there are two general limitations for using MDP in AI planning – i) absence of explanations in the MDP model and ii) inability to scale to large-size applications.

Finally, the HTN approach goes beyond classical planning by introducing task decomposition in the planning process. For example, if we wish to devise a plan to carry a passenger from London to New York, then a high-level task would be something like *Travel-from-London-to-New-York*, which could then be decomposed into lower-level tasks, such as *Book-A-Ticket*, *Go-To-Heathrow*, *Board-A-Plane-Going-To-New-York*, and *Depart-At-JFK-Airport*. The expansion procedure is described by transformation rules called *methods*. A plan is completed when the resulting network contains only primitive-tasks and a solution is consistent with the set of given constraints [5].

Although different planning paradigms provide specialized mechanisms to represent plans their main focus is on specific domains or 'planning shells' and no comprehensive formal analysis exists that provides an account of what the planning task is independent of the way the planning task may be tackled. An *ontology* [15] is a specification of the different entities and abstractions that exists in a universe of discourse for the purpose of reuse. Work on reusable components for knowledge-based systems [2, 4, 8, 18] defines *task ontologies* [18, 20] as a way of formalizing classes of generic knowledge-intensive tasks, such as parametric design, scheduling, diagnosis, independently of the way these tasks can be tackled by reasoning methods. A second class of components commonly referred to as problem-solving methods (PSMs), specify the way tasks can be solved. In this paper our main focus is to describe the planning task ontology, which formally characterizes the nature of the planning problems.

Some attempts have been made in the past [12, 14, 23, 25] at formalizing the planning task, however these formalizations suffer from limitations. In particular, none of these proposals [14, 23, 25] provide a concrete, formal ontological resource which can be used to precisely characterize planning problems. In some cases [25] the proposed models simply provide a coarse-grained analysis of the planning task. While this can be used to organize domain knowledge for a planning application, it does not provide the required degree of modeling support, which is needed to characterize a specific planning application formally and precisely. In other cases [14, 23], some degree of formalization is provided, however important concepts are missing and even when these concepts are present, they are typically under-specified. For instance, the notion of plan-task in the aforementioned approaches fails to take into consideration key aspects of a plan-task, such as time and agent. Moreover at a more general level of analysis, important aspects such as optimality concerns are not covered by these proposals.

Theoretically, our main aim is to formalize the planning task at a generic level by amalgamating different planning paradigms. Therefore, our definition of planning subsumes the characteristics of all the three planning paradigms discussed earlier. Thus we define planning as:

"A detailed formulation of a sequence of plan-tasks and actions associated with the plan-tasks. These plan-tasks and actions can be executed by agents in order to transform

[1] Given the probability distribution over the possible candidate solutions of an action in any state and a preference function over the solutions, a utility function can be defined on these candidate solutions such that whenever an agent prefers one candidate solution over another, then the preferred plan has said to exhibit a higher expected utility [3].

an initial-world-state into a goal-world-state. The output of the planning task is a ***plan*** *specified in accordance with application-specific solution criteria."*

Our ultimate aim is to develop a generic planning task ontology to overcome the shortcomings that can be observed in the existing proposals. We say that our task ontology is generic to emphasize that it is independent of specific planning domains or applications, or reasoning methods. Our ontology formalizes the planning task without subscribing to any planning specific paradigm, but tries to tease out and integrate the important conceptual and theoretical distinctions that exist in the main planning paradigms. We also aim to achieve maximum reusability by reusing the appropriate notions defined in earlier task ontologies we developed, such as parametric design [18] and scheduling [19].

Our approach is based on research in knowledge system development by reuse [8, 18], which proposes a generic epistemology for knowledge-based systems in terms of tasks, problem-solving methods and domain models, and also uses ontological engineering as the key technique for building libraries of reusable components for knowledge systems.

Our task ontology is *formalized* by using the *Operational Conceptual Modelling Language* (OCML) [18], which provides both support for producing sophisticated specifications, as well as mechanisms for operationalising definitions to provide a concrete reusable resource to support knowledge acquisition and system development. Import/export mechanisms from OCML to standards, such as OWL [17] and Ontolingua [6] ensure symbol-level interoperability.

The rest of the paper is organized as follows: the following section provides a coarse-grained specification of the planning task. In section 3, we provide a more detailed specification of the ontology. In section 4, we compare our work with existing proposals in the field and finally, in section 5, we draw some conclusions from our work and outline future research.

2 A Generic Specification of the Planning Task

In accordance with the informal definition given in the previous section, the planning task in our framework is represented as a mapping from an eleven dimensional space {S_O, G, PT, A, AG, PA, TH, C, PR, Cf, SOL} to a plan **P**. These parameters are described below:

- **Initial-world-state**, S_O. It describes the state of the world at the beginning of the planning process.
- **Goal,** G. It describes the desired state of the world we want to achieve through a planning process.
- **Plan-tasks**, PT = {pt_1, …., pt_n}. A set of *plan-tasks*, which specify intermediate goals which need to be accomplished to achieve the overall goal of the planning task.
- **Actions**. For each plan-task, pt_i, there is a finite set of actions, A_i = {a_{i1}, …., a_{ik}}, which must be executed to accomplish pt_i.
- **Agents**, AG = {ag_1, …., ag_m}. A set of agents, which are responsible for achieving plan-tasks through the execution of actions.
- **Parameters**, PA = {pa_1, …., pa_l}. Parameters can be seen as meta-level pointers to the domain entities, which are relevant to the planning process.
- **Time-horizon**, TH. A time window within which the plan is required to take place.
- **Constraints**, C = {c_1, ..., c_j}. A set of constraints, which must not be violated by a plan. Typical constraints observed in planning are *variable binding*, *ordering relation*, and *interval preservation* [26].
- **Preferences**, PR = {pr_1, …., pr_o}. A set of criteria for partially ranking competing plans. These are important to support the acquisition and modeling of local optimization criteria during the knowledge acquisition process and indeed they can in practice be mutually unsatisfiable. Preferences are typically called *soft constraints* in many ap-

proaches to design and planning, however they are ontologically very different from constraints and therefore we prefer not to use the term "soft constraint". We will discuss this point in more detail in section 3.6.

- **Cost-function**, Cf. A function, which provides a global mechanism for comparing the costs of alternative plans.
- **Solution criterion**, SOL. A mapping from a plan **P** to {True False}, which determines whether a candidate plan is a solution. A solution criterion usually requires **P** to be *complete* and *valid* - see the following section for the description of these properties.
- **Plan-model**, P = {p_1, …., p_q}. A candidate plan is a sequence of pairs, <pt_i, ag_j>, where pt_i is a plan-task and ag_j is an agent able to execute the relevant actions associated with pt_i.

Solution criteria tend to be application dependent. For instance one application may require optimal solutions, while another one may require simply valid solutions, which satisfy all applicable constraints. Thus the ontology provides modular definitions to allow application builders to assemble specific solution criteria from reusable components as well as some default generic applicable criteria. In particular, two key notions are *completeness* and *validity*, which are specified as follows:

- A plan model **P** is ***complete***, if every plan-task in PT is included in P.
- A plan model **P** is ***valid***, if it does not violate any of the plan-constraints in C. It is important to emphasize that in contrast with classical planning, constraints may not only impose conditions on what must be true in the goal state, but they might have to be satisfied by all the intermediate states, traversed during the planning process.

3 A Formal Specification of the Task Ontology

Our task ontology consists of 87 definitions and it relies on two other ontologies: Simple Time ontology and Base Ontology. The Simple Time ontology was imported from the Ontolingua library and is based on Allen's [1] theory of time. The Base Ontology provides support to define standard modeling definitions, such as tasks, relations, functions, etc. The complete specification of the ontology can be found at URL http://kmi.open.ac.uk/people/dnyanesh/ontologies/planning.

3.1 Planning-Task and Plan-Solution

The starting point for understanding the ontology is the class `planning-task`, which characterizes the planning problem in terms of *input and output roles*, *precondition*, and *goal-expression*, consistently with our modeling framework based on tasks and PSMs [8, 18]. Because we have already discussed the input and output roles to a planning task in Section 2, here we only need to discuss the precondition and goal-expression of the task. The *precondition* states that the planning task must include a description of the initial and goal states, and there has to be at least one plan-task for the meaningful generation of a plan. The *goal-expression* states that the goal of the planning task is to devise a plan according to the given solution criterion. If none is provided then the default one is applied (cf. Section 2), which is modeled by the relation `plan-solution`.

3.2 Initial State and Goal Description

Both the initial state and the goal state descriptions are simply sets of statements which express the current state of the world and the desired state we wish to achieve. These state-

ments express properties of parameters. Formally a goal description is a logical expression with no free variables.

3.3 Plan Task

Plan tasks are represented in terms of a number of attributes:

Has-parameters: this slot represents the parameters associated with a plan-task. Strictly speaking this slot is not necessary – the relevant parameters can be identified simply by analyzing the precondition, actions and postcondition of a plan, however it is useful to be able to associate parameters to plan tasks directly, to simplify the reasoning process.

Has-precondition: this slot contains a logical statement, which states what must be true before a plan-task is executed. For instance, if a plan-task is to perform a welding operation by a robot, then the robot's arm must be holding a welding gun before performing the welding operation.

Has-postcondition: this slot contains a logical statement, which states what must be true after the execution of a plan-task.

Achieved-by-actions: this slot indicates the actions which need to be executed to achieve the plan task. For instance, if a plan-task is to *Book-An-Air-Ticket* then actions, such as *Find-A-Travel-Agent*, *Check-The-Ticket-Availability*, *Check-The-Mode-Of-Payment*, etc. are needed to achieve the booking.

Requires-agents: this slot associates a plan-task to one or more agents able to achieve the plan task.

Has-time-range: this slot represents a time-window within which a plan-task must take place. It is represented by the earliest and the latest start and end times.

The box below shows the OCML definition of the class.

```
(def-class PLAN-TASK () ?pt
 ((has-parameters :type parameter :min-cardinality 1)
  (has-precondition :type relation-expression
                    :default-value (true))
  (has-postcondition :type relation-expression)
  (achieved-by-actions :type action
                       :min-cardinality 1)
  (requires-agents :type agent :min-cardinality 1)
  (has-time-range :type job-time-range :max-cardinality 1)))
```

In our task ontology various relations and functions can be used to check whether a plan-task is achieved successfully while constructing a plan. The relation `agent-executes-plan-task (agent plan-task plan-model)` is a ternary relation between an agent, a plan-task, and a plan-model, states that an agent executes the plan-tasks in a planning world. The relation `plan-task-is-achieved (plan-task plan-model)` is a binary relation between a plan-task and a plan-model, checks whether a plan-task is achieved by the agents through the execution of the actions associated with it. The relation `plan-task-is-unachieved` is the inverse of relation `plan-task-is-achieved`. The relations in our framework are important to capture the control and mechanism input [10] for accomplishing plan-tasks to construct a solution plan. The functions `actions-associated-with-plan-task` and `agents-associated-with-plan-task` are respectively used to retrieve all the actions and agents associated with a plan-task while constructing a plan. Finally the function called `duration-of-a-plan-task` is defined to calculate the duration of a plan-task. This function takes as an input the time range of a plan-task and calculates the plan-task duration by subtracting the end time of a plan-task time range from the start time of a plan-task time range.

3.4 Action

Each action is associated with a plan-task, say pt_i, and is also defined in terms of a precondition and a postcondition. The main difference between actions and plan tasks is that the latter decompose into a number of actions while an action is not further decomposed in a planning model. For instance, in transportation planning, if a plan-task is to *Load-Package-Into-Vehicle* then some of the actions associated with this plan-task are *Open-Door*, *Perform-Package-Loading*, *Close-Door*, etc. The box below shows the OCML definition of the class.

```
(def-class ACTION () ?a
 ((has-precondition :type relation-expression :default-value (true))
  (has-postcondition :type relation-expression)))
```

Similarly to the Process Interchage Format [10], in our task ontology we define a number of ordering relations that can be used to impose temporal ordering constraints among any two plan-tasks and actions: `BEFORE`, `FOLLOWS`, `MEETS`, `OVERLAPS`, `DURING`, `START-SIMULTANEOUSLY`, `FINISH-SIMULTANEOUSLY`, and `START-AND-FINISH-SIMULATENOSLY`.

3.5 Agent

The class `agent` represents a person, group, or an entity which holds a purpose and is responsible for achieving different plan-tasks through the execution of actions. It has the following attributes:

Can-Executes-plan-task: this slot associates each agent to all the possible plan-tasks it can execute to construct a plan.

Holds-a-purpose: this slot expresses the cognitive attitude of an agent. According to the Suggested Upper Merged Ontology [22], the purpose of an agent can be represented by a *physical entity*, *another agent* (cognitive or sentient), or a *formula*.

Has-time-window: this slot represents a limited time-window within which an agent is available to accomplish plan-tasks. It is represented in terms of a start and an end time.

The box below shows the OCML definition of class `agent`.

```
(def-class AGENT () ?ag
 ((can-executes-plan-task :type plan-task)
  (holds-a-purpose :min-cardinality 1)
  (has-time-window :type time-range :max-cardinality 1)))
```

The class `agent` is specialized into `sentient-agent` and `cognitive-agent` [22]. The former represents an agent that has rights, but may or may not have the ability to reason, while the latter is a sentient agent that has ability to reason, deliberate, or make plans, e.g. human-beings.

3.6 Constraint and Preference

Although, some of the existing proposals [23, 25] fail to distinguish between the notions of constraints and preferences, our task ontology provides a clean distinction between them, and provides a set of possible validation criteria to validate a candidate plan. The class `constraint` represents a property that must not be violated by a valid solution. One of the earlier planning frameworks like I-N-OVA [24] determines plan validity based on specific constraint types, such as *ordering*, *variable*, and *auxiliary* constraints. In contrast with this approach, our framework does not limit plan validation only to specific types of constraints, but rather allow users to specify constraints according to their own applications.

Because our task ontology does not subscribe to any specific classification of constraints it allow users to specify different types of constraints, such as theoretical, technical, engineering, social or cognitive control actions, which can be used to determine the space of a valid plan. Usually the social and cognitive constraints can be seen as the properties that emerge from the environment, which is external to the core planning world. For instance, if we wish to construct a plan for an automotive industry that manufactures cars, then one essential social constraint that must be taken into account is to ensure that no human-resources work more than the legal shift hours. In comparison with other types of constraints, the cognitive constraints are different in nature because they can be used during planning to reason about the cognitive abilities of the agent performing certain tasks. For example, if a planner has to come up with a plan to design a car then he/she must take into account the cognitive ability of the average human-being in dealing with car mechanism.

As described earlier (cf. Section 2), the notion of a soft constraint is typically used to determine the criteria associated with an optimal solution, and accordingly a solution that satisfies a maximum number of soft constraints is treated as a better solution than other competing solutions. In contrast with this, in our model constraints define *prescriptive* properties because they must not be violated by the plan-tasks and associated actions. If soft constraints are allowed to be violated while constructing a solution [27], then it is not clear how they can be used as a prescriptive measure to determine the space of valid solutions. For this reason we prefer to use the notion of `preference`, which we believe more clearly expresses the ontological role necessary to model the plan optimization criterion in different planning applications. The preferences are essentially used to describe the choice points available to a planner while constructing a plan. Logically a constraint maps a plan model to {True, False}. In contrast with constraints, preferences express a partial ranking of plans.

3.7 Plan Optimization Criterion

A planning task may be concerned not only with finding a complete solution that satisfy constraints, but also with trying to optimize over some criterion. Our task ontology captures a plan optimization criterion, which chooses an optimal plan which has the least cost according to some evaluation function, e.g., minimization of idle time.

Optimization is handled by means of two constructs, which capture the knowledge needed to rank candidate solutions: *preference* and *cost-function*. The cost-function provides a global optimization criterion and must be consistent with the various partial rankings provided by preferences. To this purpose the ontology includes two axioms, `cost-subsumes-preferences` and `cost-preference-consistency`. The first axiom states that the cost-function should enforce the partial order expressed by all the relevant preferences. The second axiom states that the cost-function should not violate any preferences. Because preferences tend to be heterogeneous in nature, the cost-function may not only be a numeric value but some non-Archimedean criterion [18] may be needed. The box below shows the OCML definitions of the plan optimization criterion.

```
(def-relation OPTIMAL-PLAN (?pl-1 ?task)
 :constraint (and (planning-task ?task)
                  (plan-model ?pl-1))
 :iff-def (and (plan-solution ?pl-1 ?task)
               (not (exists ?pl-2
                            (and (plan-solution ?pl-2 ?task)
                                 (has-cost-order-relation ?task ?rel)
                                 (cheaper-plan ?rel ?pl-1 ?pl-2))))))
```

```
(def-relation CHEAPER-PLAN (?rel ?pl-1 ?pl-2)
 :constraint (and (order-relation ?rel)
                  (plan-model ?pl-1) (plan-model ?pl-2))
 :iff-def (holds ?rel ?pl-1 ?pl-2))
```

3.8 Axioms in the Task Ontology

A number of axioms are included in the task ontology, which precisely circumscribe the set of models consistent with the ontology and can be used to ensure consistency during the planning process:

- **Plan-Task-Ordering-Maintains-Pre-And-Post-Conditions.** This axiom states that if two plan-tasks are constrained by an ordering relation, such as (FOLLOWS PT_2 PT_1), then the precondition of PT_2 must hold after the execution of PT_1. This axiom is particularly important to deal with the *conflict exclusion* condition [26].
- **Plan-Maintains-Complete-Exclusion.** This axiom maintains a complete exclusion among any two plan-tasks throughout the plan construction. It states that no two plan-tasks can occur at the same time if they are consuming the same agent [26].
- **Condition-Consistency-Among-Plan-Task-And-Action.** This axiom states that, if there exists a plan-task, say pt_i, which has a sequence of actions associated with it, say $\{A_{i1}, \ldots, A_{in}\}$, then the precondition of pt_i must hold for the first action, A_{i1}, to be executed and the postcondition of pt_i must hold after the execution of the last action, A_{in}. The box below shows the OCML definition of this axiom.

```
(def-axiom CONDITION-CONSISTENCY-AMONG-PLAN-TASK-AND-ACTION
 (forall (?pt)
         (=> (plan-task ?pt achieved-by-actions ?actions)
             (and (= (first ?actions) ?a1)
                  (element-of (?pt . ?ag) ?pl)
                  (plan-model ?pl)
                  (agent ?ag)
                  (= (the ?x1 (has-precondition ?a1 ?x1)) ?a1-pre)
                  (= (last ?actions) ?a-n)
                  (element-of (?pt . ?ag) ?pl)
                  (= (the ?x2 (has-postcondition ?a-n ?x2)) ?an-post)
                  (holds (the ?pt-pre (has-precondition
                                       ?pt ?pt-pre)) ?a1-pre)
                  (holds (the ?pt-post (has-postcondition ?pt ?pt-post))
                         ?an-post)))))
```

4 Related Work

The notion of a plan in the PLANET ontology [14] is represented by the following key concepts: initial-planning-context, goals, actions, tasks, and choice-points. The initial-planning-context represents the initial, given assumptions about the planning problem. It describes a background scenario in which plans are constructed and must operate on. The initial-planning-context consists of an initial state, goal description, and external constraints and a resulting plan is represented as a set of commitments to actions taken by an agent to achieve the specified goals. A world state in the PLANET ontology represents a model of the environment for which a plan is designed. A certain world state can be chosen as an initial state for a given planning problem. All the solution plans must take into account this initial state. The goals describes what needs to be achieved in the process of constructing a plan, however the initial planning context may not directly specify the goals to be achieved but can be deduced from the initial information about the situation and the constraints that are imposed on a planning problem. A solution plan is validated against completion, con-

sistency, feasibility, and justified criteria. The notion of a choice point represents various choices typically needed while devising a plan, such as task commitment, ordering commitment ($Action_1$-BEFORE-$Action_2$), etc. Some crucial differences exist between the PLANET ontology and our task ontology. The current version of the PLANET ontology does not include some key notions, such as agents, resources, time, and location. Moreover the goal of a planning task may not just be to find a valid solution, but may also require some optimization criterion. Because the notion of a cost function is absent in the PLANET ontology, these issues cannot be modeled in this approach.

The plan ontology [23] takes a top-down approach to ontology construction. One of the main aims of the plan ontology is to provide a conceptual framework which can form the basis for more in-depth analyses and as a result the level of detail of the definitions in the plan ontology is very coarse-grained. For instance, the concept *activity* (which is analogous to the concept of a plan-task in our framework) is defined in terms of i) the *agent* required to execute the activity, ii) the *time range* of the activity, and iii) the *temporal constraints* among activities. Our ontology provided a much more fine-grained characterization of plan tasks, supporting the specification of actions, preconditions and postconditions. Consistently with the classical planning approach [9], the definition of activity in the plan ontology does not take into account the duration of an activity. As pointed out in [11], if the aim of the planning task is simply to sequence all the activities, then the absence of duration does not create much problem. However, duration is needed in order to handle the overlapping of activities. Moreover, as described in Section 3.5 in our framework the notion of an `agent` is specialized into `sentient-agent` and `cognitive-agent` on the basis of its cognitive behavior, but no such kind of specialization is considered in the plan ontology. Another difference between the plan ontology and our task ontology is that while the former takes into account the notion of a preference as the agent's *desire*, the plan ontology does not provide any indication about how these *desires* affect the cost of a plan, to the extent that this information can be used to optimize a solution. In contrast with their framework, our task ontology handles plan optimization (cf. Section 3.7) by including support for modeling the notions of cost and preference. Finally, our task ontology provides a number of different axioms (cf. Section 3.8), which allow us to formalize the general (i.e., application-independent) conditions, which must be maintained during plan construction.

Valente [25] provides a knowledge-level analysis of the classical planning task. The fundamental difference between our task ontology and Valente's framework is that while the main aim of his framework is to analyze the *classical planning task*, no such kind of commitment is assumed while formalizing the planning task in our task ontology. The main aim of our task ontology is to combine the important conceptual and theoretical distinctions that exist in different planning paradigms. More importantly, Valente's framework uses the notion of a soft constraint to rank competing solution plans, but, as discussed in section 3.6, we find the notion of a soft constraint problematic, and instead we prefer to use the notion of preference, which we believe provides a more systematic way to rank competing plans to determine the optimal solution. In contrast with our task ontology (cf. Section 3.7) no optimization criterion is considered in Valente's framework. Moreover, in contrast with the informal analysis of the planning task proposed by Valente, our task ontology is formalized in detail.

The DDPO framework [12] takes as an input concepts and relations specified in DOLCE along with some of its extensions, notably the Ontology of Descriptions and Situations (D&S) [13]. Similarly to our task ontology, all the concepts, relations, and axioms in the DDPO framework are specified formally, and therefore, this ontology provides a high level of formalization. The primary aim of DDPO is to formalize the plans at an abstract level and independently from its existing resources. In line with D&S, DDPO also includes physical and non-physical objects, events, states, regions, qualities, and constructivist situa-

tions. Central to DDPO is the notion of tasks, the types of actions, the sequencing among tasks and actions, and the controls performed on them. The domain specific tasks are clearly distinguished from the actions and states, and the control operators of classical planning are considered as types of planning actions of different nature from executable actions. Although the DDPO framework specializes the notion of a task into complex task, sequential task, hybrid task, bag task, and action task, it is not clear how the duration of each task is represented to handle the task overlapping [11]. Similarly to our task ontology, other procedural actions from classical planning, such as precondition, postcondition, and preferences are also taken into account. A plan is a description that uses at least one task and one agentive role to achieve a goal. A goal is a desire that is part of a plan and is satisfied by a goal situation. Similarly with our task ontology, DDPO validates a solution plan against satisfaction of accompanying conditions (which is referred to as a constraint (cf. Section 3.6) in our framework), which are specified as a relation between a situation and a task. In contrast with our task ontology, the DDPO framework does not talk about different axioms that are necessary to deal with different conditions in planning such as *conflict exclusion* [26] and *complete exclusion among plan-tasks* [26]. Finally, in line with our task ontology (cf. Section 3.7) DDPO handles plan optimization in terms of preferences and cost function although the cost functions are not definable within DDPO ontologies.

5 Conclusion and Future Work

In this paper we have described our initial version of a generic planning task ontology. We say that our task ontology is generic to emphasize that it is independent of specific planning domains, applications, or reasoning methods. Because our task ontology does not subscribe to any particular problem-solving approach, it provides a generic foundation that can be adopted by the different reasoning services. Moreover, it provides us with a strong engineering tool for knowledge acquisition and to represent different planning applications.

The next step will be to evaluate the ontology on a number of planning applications to confirm its generic nature and verify its modeling value. We are also planning to use this task ontology as the ontological basis for constructing problem-solvers for planning, both to provide a concrete library of reusable reasoning methods for planning and also to develop novel insights into the planning process.

Acknowledgements

The authors would like to thank the anonymous referees of the FOIS'04 conference for their valuable comments, which helped us to improve the quality of the paper.

References

[1] James Allen. Maintaining Knowledge about Temporal Intervals. Communications of the ACM, 26 (11): 832-843, 1983.

[2] V. Richard Benjamins. Problem Solving Methods for Diagnosis. PhD Thesis, University of Amsterdam, Amsterdam, The Netherlands, 1993.

[3] Jim Blythe. Decision-Theoretic Planning. AI Magazine, 20 (2), 1999.

[4] Balakrishnan Chandrasekaran. Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert Systems Design. IEEE Expert, 1 (3): 23-30, 1986.

[5] Kutluhan Erol, James Hendler, and Dana S. Nau. Semantics for Hierarchical Task-Network Planning. Technical report CS-TR-3239, University of Maryland at College Park, 1994.

[6] Adam Farquhar, Richard Fikes, and James Rice. The Ontolingua Server: a Tool for Collaborative Ontology Construction. International Journal of Human-Computer Studies, 46 (6): 707-728, 1997.

[7] Eugene A. Feinberg and Adam Shwartz (editors). Handbook of Markov Decision Processes: Methods and Applications. International Series in Operations Research and Management Science. Kluwer Academic Publishing, 2001.

[8] Dieter Fensel and Enrico Motta. Structured Development of Problem-Solving Methods. IEEE Transaction Knowledge and Data Engineering, 13 (6): 913-932, 2001.

[9] Richard Fikes and Niles J. Nilson. STRIPS: A New Approach to the Application of Theorem Proving and Problem Solving. Artificial Intelligence, 2: 189-208, 1971.

[10] Jintae Lee, Michael Gruninger, Yan Jin, Thomas Malone, Austin Tate, Gregg Yost, and other members of the PIF Working Group. The PIF Process Interchange Format and Framework. Working Paper Series/ MIT Center for Coordination Science, 1996.

[11] Maria Fox and Derek Long. PDDL+: An Extension to PDDL to Handle Time. Technical Report, Durham University, 2001a.

[12] Aldo Gangemi, Stafano Borgo, Carola Catenacci, and Jos Lahmann. Task Taxonomies for Knowledge Content. METOKIS Deliverable, D07, 2004.

[13] Aldo Gangemi and Peter Mika. Understanding the Semantic Web Through the Descriptions and Situations. In Proceedings of the DOA/CoopIS/ODBASE Confederated International Conferences DOA, CoopIS, and ODBASE, LNCS, Springer, 2003.

[14] Yolanda Gil and Jim Blythe. PLANET: A Sharable and Reusable Ontology for Representing Plans. In Y. Gil and K. L. Myers, editors, Proceedings of the AAAI - Workshop on Representational Issues for Real-World Planning Systems, pages 28-33, 2000.

[15] Thomas R. Gruber. A Translation Approach to Portable Ontology Specification. Knowledge Acquisition, 5 (2): 199-221, 1993.

[16] Allan Newell and Herbert A. Simon. GPS: A Program that Simulate Human Thought. In E. A. Feigenbaum and J. Feldman, editors, Computers and Thought: 279-293, 1963.

[17] Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. W3C Recommendation, 10 February, 2004, http://www.w3c.org/TR/2004/REC-owl-guide-20040210/, 2004.

[18] Enrico Motta. Reusable Components for Knowledge Modelling - Principles and Case Studies in Parametric Design Problem Solving. IOS Press, The Netherlands, 1999.

[19] Enrico Motta, Dnyanesh Rajpathak, Zdenek Zdrahal, and Rajkumar Roy. The Epistemology of Scheduling Problems. F. V. Harmelen, editor, Proceedings of the 15th European Conference on Artificial Intelligence, Lyon, France, pages 215-219, 2002.

[20] Riichiro Mizoguchi, Johan Vanwelkenhuysen, and Mitsuru Ikeda. Task Ontology for Reuse of Problem Solving Knowledge. Towards Very Large Knowledge Bases, pages 46-57, IOS Press. 1995.

[21] Dana S. Nau, Yue Cao, Amnon Lotem, Hector Munoz-Avila. SHOP: Simple Hierarchical Ordered Planner. T. Dean, editor, Proceedings of the 6th International Joint Conference on Artificial Intelligence, Stockholm, Seden, pages 968-975, 1999.

[22] Ian Niles and Adam Pease. Towards a Standard Upper Ontology. Proceedings of the International Conference on Formal Ontology in Information Systems, Ogunquit, Maine, USA, pages 2-9, 2001.

[23] Austin Tate. Plan Ontology - a Working Document. In Proceedings of the Workshop on Ontology Development and Use, Le Jolle, CA. 1994.

[24] Austin Tate. Representing Plans as a Set of Constraints - the <I-N-OVA> Model. B. Drabble, editor, In Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems, Edinburgh, Scotland, pages, 221-228, 1996.

[25] Andre Valente. Knowledge-Level Analysis of Planning Systems. SIGART Bulletin, 6 (1): 33-41, 1995.

[26] Daniel S. Weld. Recent Advances in Planning. AI Magazine, 20 (2): 93-123, 1999.

[27] Mark Zweben and Mark S. Fox. Intelligent Scheduling. Morgan Kaufmann, San Francisco, California, USA. 1994.