An Ontological Formulation and an OPM profile for Causality in Planning Applications

Irene Celino and Daniele Dell'Aglio

CEFRIEL – Politecnico of Milano, Via Fucini 2, 20133 Milano, Italy
{irene.celino,daniele.dellaglio}@cefriel.it

Abstract. In this paper, we propose an ontological formulation of the planning domain and its OWL 2 formalization. The proposed metamodel conceptualizes planning rules and actions and the *causality* between them. We also show that our planning metamodel can be seen as a relevant scenario of the Open Provenance Model (OPM) and we define our planning OPM profile.
This ontological representation is then exploited to define automated means for the verification of correctness and consistency of a planning domain model. We claim that Semantic Web technologies can provide an effective solution to this important – and often underestimated – problem for planning applications.

1 Introduction and Motivations

Planning is a branch of AI dealing with the automated creation of plans, i.e. a sequence of actions that, starting from an initial state of the "world", leads to a state that meets some desired goals. Finding a suitable plan among all possible sequences of actions is a complex problem that requires a detailed comprehension of the world, the agents acting in this world, the possible actions, the causal relationships between those actions, etc. The specification of such knowledge about the planning problem is usually indicated as *domain theory*.

When formalizing a domain theory in some representation format, it is therefore key to use an enough expressive language and to assure the correct and consistent representation of the world, so that the planning algorithms can operate and compute the optimal plans. Several approaches and languages have been proposed in literature to formalize the planning problem, including the well-known STRIPS [1] and the more recently standardized PDDL language [2]. Those representation formats and languages are also employed to check the consistency of the generated plans (e.g., to recognize an inconsistency in a plan if two incompatible events are applied simultaneously).

Checking the coherence and rationality of the domain theory itself, on the other hand, is a task that is usually discarded or delegated to the modeller that formalize the planning problem. When modelling the actions and their causal relationships, for example, it is indeed important to validate the model, e.g. by checking that all modelled states of the world are "reachable" in a sequence of actions. Especially when the domain theory is large and complex, this kind of validation becomes of utmost importance.

In this paper, we illustrate how the Semantic Web can be successfully employed to represent and reason upon such planning domain models. We propose an ontological formalization of a planning metamodel, i.e. a domain-independent specification of the planning problem, on top of which different domain theories can be developed to represent different planning situations. In doing so, we also show that the planning metamodel can be seen as a relevant application case of the Open Provenance Model (OPM) [3]; we provide an OPM profile by mapping the planning metamodel to one of the ontological formulation of OPM, namely the Open Provenance Model Vocabulary (OPMV) [4]. The ontological formulation can then be employed to define a set of rules to check the consistency of a domain theory that uses the planning metamodel; this is aimed at giving modellers a means to check and support their modelling task.

The remainder of the paper is structured as follows. Section 2 introduces the ontological formulation of the planning problem, together with its representation in OWL 2 [5] and the explanation of our modelling choices; Section 3 explains the planning metamodel as an OPM Profile and its semantics; Section 4 gives some examples of automated checks of the causality in domain theories defined on the planning metamodel. We present a complex scenario of Simulation Learning for Crisis Management in Section 5, in which we apply our metamodel; related work is illustrated in Section 6 while Section 7 concludes the paper and gives some hints on possible extensions of this work.

2 The Ontological Formulation of the Planning Problem

Ontologies are generally used to *assert* statements that are considered true in the modelled world; in planning application, however, dynamics is the predominant dimension, e.g., in a planning domain theory we can include definition of actions that are mutually exclusive and therefore cannot be "asserted" at the same time. Thus it could be considered unusual on unsuitable to use ontologies to represent planning knowledge.

However, our investigation does not deal with the search for the optimal plan, i.e. our formalization does not want to be used within the planning algorithm. Indeed, our purpose is different: through an ontological representation, we aim to predicate on the possible states of the planning world and on the causality of state transitions, i.e. the conditions under which the world state changes. Within the planning metamodel we want to "statically" represent the possible "dynamics" of the world.

In this section, we explain our modelling of the planning problem and of its causality definition. We reuse as much as possible the terminology used in planning literature [6, 1, 2]; some part of the vocabulary can specifically refer to the terminology used in timeline-based planning [7, 8].

2.1 Planning metamodel

As introduced above, a *domain theory* defines the planning problem, in terms of the agents, their possible actions and the causality between them. The *planning*

problem consists in identifying a set of relevant aspects whose (temporal) evolutions need to be controlled to obtain a desired behaviour or to attain a specific goal. In the following, we introduce the main primitives of a domain theory.

Components represent logical or physical subsystems whose properties may vary in time, thus they are the relevant variables to be planned for. Components evolve over time; events can happen and decisions can be taken on components: those events and decisions alter the components evolution. Components can either be intelligent *agents*, i.e. the characters involved in the planning – both human and artificial ones –, or other entities, such as buildings, weather, rivers, which are *resources* to be controlled in the planning.

We can classify components on the basis of their *control*. The temporal behaviour of *controllable components* is decided by the planner; those components define the search space for the planning problem, and their evolution ultimately represent the problem solution. Conversely, the evolution of *uncontrollable components* is given to the planner as input; those components are imposed over time thus they can only be observed: they can be seen as additional/external data and constraints for the planning problem.

Actions are temporally tagged events. They are always related to a component and they represent events or decisions that alter the component behaviour, causing the transition of the component between two possible states. Usually the term *decision* is used in relation to an uncontrollable component, while the term *event* is preferred to indicate an action determined by the planner that happens to a controllable component; in the following we will use only the term action for the sake of simplicity. Actions always refer to some component and are characterized by some *parameters*. The conditions under which actions can occur are regulated by the domain theory.

The domain theory defines *actions' causality*, i.e. the combinations of components behaviours that are acceptable with respect to actions happening on other components. We represent such causality by means of *planning rules*, also called *synchronizations*. A planning rules specifies the "consequences" of actions, i.e. it states the relation between two or more actions. In their generic form, planning ruless relate how an action (also called reference action) can activate one or more actions (target actions) if some conditions or constraints on components and/or on action's parameters are verified.

Rule conditions are the constraints defined within a planning rule. In their generic form, rule conditions impose requirements on the actions involved in a synchronization, thus they can be represented as relations between action parameters. A special case of constraint, very relevant for the timeline-based planning problem, is the *temporal condition*: it represents a temporal constraint on an action (e.g., an action must have a duration of 10 minutes) or on the temporal sequence of two actions (e.g., an action must start only when another action finishes). Allen's Interval Algebra [9] is used to formalize temporal conditions.

Let's consider the following example of planning rule: if an ambulance with a patient arrives at a hospital and the latter has remaining capacity, then the patient can be admitted and the number of available beds is decreased by one.

In the example we identify two components, the ambulance and the hospital; a reference action (the ambulance carrying the patient to the hospital), a target action (the hospital reserves a bed to the patient) and a condition (there are available beds in the hospital). It is important to note that a reference action can enable the activation of one or more actions (if the rule conditions are satisfied).

2.2 An OWL 2 formulation of the planning metamodel

We represent the conceptualization defined in the previous section in OWL 2 [5]. The complete definition of this ontology is available at `http://swa.cefriel.it/ontologies/tplanning`[1]; in this section, we explain some of the modelling choices.

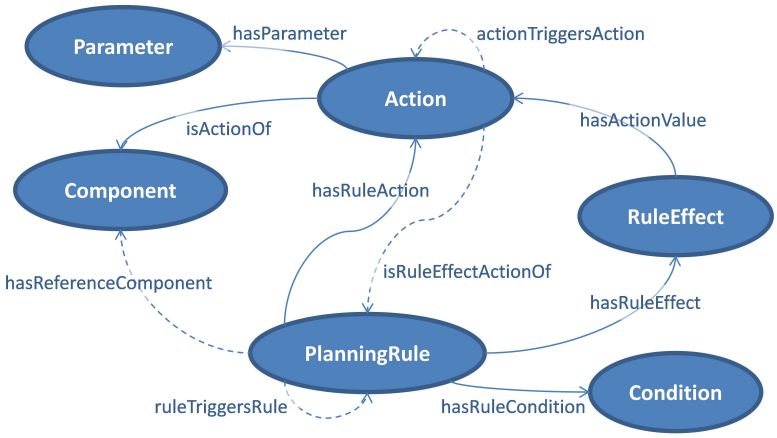

Fig. 1. A graphical representation of the main entities in our planning ontology

Figure 1 illustrates the concepts introduced above with the properties that interrelate them. `Actions` refer to `Components` and are described by `Parameters`; a `PlanningRule` puts in relation some causal action (related via the `hasRuleAction` property) with some `RuleEffect` (which, in turn, can include "target" actions); the causality constraints in a planning rule are defined by `Conditions`.

With the use of property chain axioms [5], we introduce also some derived properties. For example, starting from the basic properties defining a planning rule (indicated by a solid line in Figure 1), we infer other properties that identify the planning causality (indicated by a dashed line), as follows:

$$\begin{aligned}
\text{tpl:hasReferenceComponent} &\sqsubseteq \text{tpl:hasRuleAction} \circ \text{tpl:isActionOf} \\
\text{tpl:isRuleEffectActionOf} &\sqsubseteq \text{tpl:hasActionvalue}^- \circ \text{tpl:hasRuleEffect}^- \\
\text{tpl:actionTriggersAction} &\sqsubseteq \text{tpl:hasRuleAction}^- \circ \text{tpl:isRuleEffectActionOf}^- \\
\text{tpl:ruleTriggersRule} &\sqsubseteq \text{tpl:isRuleEffectActionOf}^- \circ \text{tpl:hasRuleAction}^-
\end{aligned}$$

[1] In the following, the `tpl` prefix (*t*emporal-*pl*anning) will be used to indicate terms from this ontology. We omit the `tpl` prefix in the figures for sake of readability.

The last two properties indicates the *causality between actions* (an action triggers another action if there is a planning rule which is activated by the first action and has the second action as effect of its activation) and the *causality between rules* (a rule triggers another rule if there is an action which is an effect of the first rule and activates the second rule).

2.3 Modelling conditions in planning rules

Most part of the planning causality, however, is usually included in the rule conditions. As introduced above, a condition imposes restrictions on the involved actions, their components or their parameters. Conditions can be assignments (in the example above, the hospital capacity is decreased by one), constraints (e.g., the hospital capacity must be greater or equal to one) or temporal conditions (e.g., the ambulance must arrive at the hospital before the patient can be admitted). Figure 2 illustrates the different types of rule conditions.

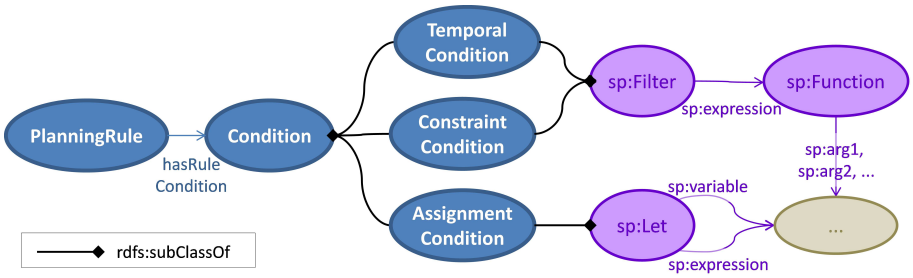

Fig. 2. Modelling of conditions in planning rules and their relation to SPARQL clauses (FILTER and LET) as modelled in SPIN.

In the SPARQL query language [10], FILTER clauses are used to express constraints on query variables; moreover, even if not part of the official specification, some SPARQL extensions also define LET clauses to express assignments on query variables[2]. It was therefore natural to us to assimilate planning rule conditions to SPARQL FILTER and LET clauses. For their modelling, we reused the well-known SPARQL Inferencing Notation [11], also known as SPIN (see again Figure 2). SPIN allows to model SPARQL queries in RDF, thus enabling to define query patterns together with the vocabulary or ontology they refer to. In our case, SPIN allows us to model the conditions that affect planning actions and their parameters together with the definition of the actions themselves.

SPIN already defines a number of different constraint types, called *functions*; for example, SPIN models Boolean functions (equal, not equal, greater than, etc.), mathematical functions (addition, subtraction, etc.) and operations on strings (e.g. regular expressions). Since in planning the time dimension has an important role in the causality definition, we extended this SPIN modelling of

[2] For example, the popular Jena framework implements the LET clause in the ARQ library.

functions to include *temporal relations* as defined by Allen [9]. Figure 3 shows our modelling: we defined a `TemporalFunction` for each Allen relation (before, after, temporal-equal, etc.); each function can be further described by one or more temporal "ranges", that indicate the intervals characterizing the relation (e.g. action A starts 5 to 10 minutes after action B ends).

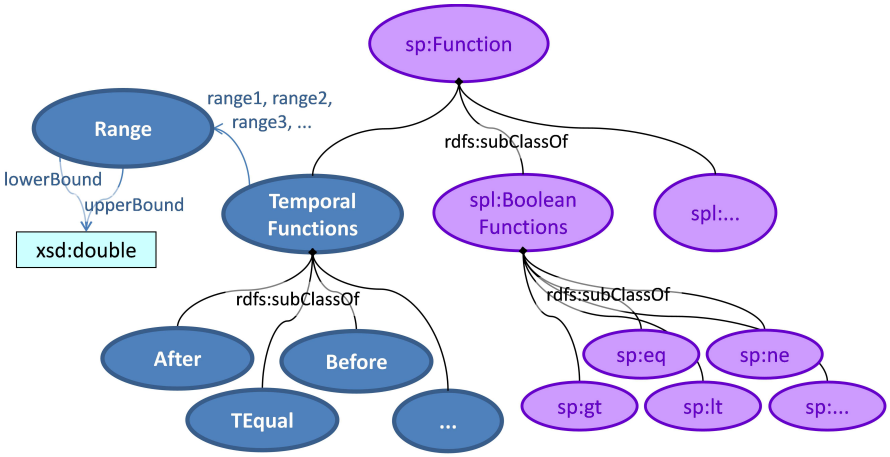

Fig. 3. Extension to the SPIN modelling of functions with Allen's temporal relations.

3 The Planning Causality as an Open Provenance Model

The Open Provenance Model Specification [3] was designed to meet several requirements, among which defining provenance in a technology-agnostic manner, supporting a digital representation of provenance and defining a core set of rules that identify the valid inferences that can be made on provenance representations. The basic nodes and edges in OPM are graphically represented in Figure 4.

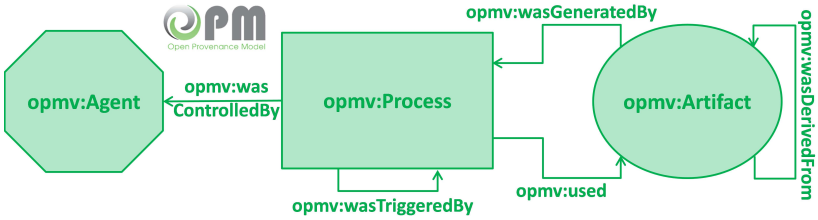

Fig. 4. The basic constituents of a provenance graph according to OPM.

According to this specification, an OPM profile is a specialisation of OPM that remains compatible with the semantics of OPM but that defines a best practice or usage guideline for a specific provenance application. The planning metamodel introduced in Section 2 can be seen as an OPM profile, in that the causality between actions by means of planning rules is a way to represent the actions

"derivation", "use" and "generation" during the planning process. It is worth noting that, while OPM is usually employed to trace provenance in *past* process executions, in the planning case the actions causality represents a *potential future* provenance information (indeed, the actions defined in a planning domain model represent action "templates" rather than "instances" of actions).

In this section, we define our Planning OPM profile and its formalization in a mapping between our planning metamodel and the Open Provenance Model Vocabulary [4], one of the ontological formulations of OPM. We also express the OPM completion rules and inferences, as defined in [3], and we explain how we relaxed some of the OPM constraints to better capture the concept of "provenance" in planning. We are aware that the formalization provided in the following has stronger assertions than those in OPM; still, we confine those restrictions to our OPM profile, in which they are meaningful and valid, and do not intend them as of general value outside our profile.

3.1 Mapping the planning metamodel to OPMV

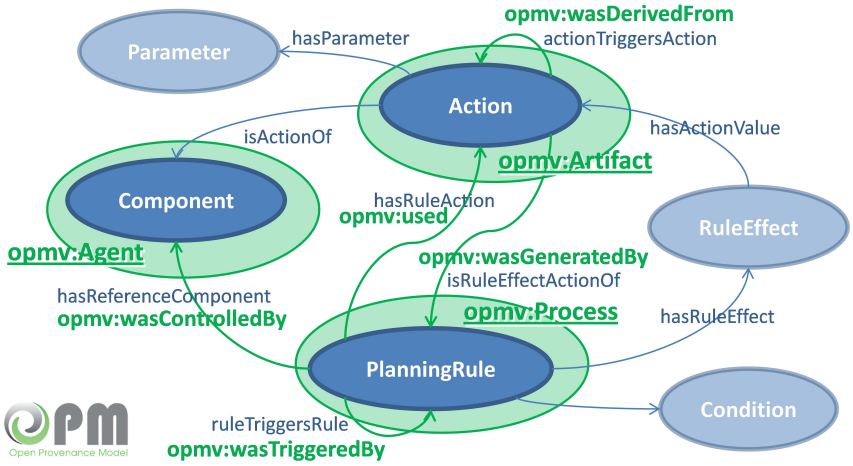

Fig. 5. The planning metamodel as an OPM Profile.

Figure 5 graphically shows our planning OPM profile[3] (cf. with Figure 1). The planning rules are our main *processes*, the components are our *agents* and the actions are the *artifacts* of the planning. Thus, with reference to the OPM nodes and edges as defined in the OPM Vocabulary [4], in our planning profile we assert that:

$$\begin{aligned} \text{tpl:Component} &\sqsubseteq \text{opmv:Agent} \\ \text{tpl:PlanningRule} &\sqsubseteq \text{opmv:Process} \\ \text{tpl:Action} &\sqsubseteq \text{opmv:Artifact} \end{aligned}$$

[3] The complete definition of the planning OPM profile is also available on the Web at `http://swa.cefriel.it/ontologies/causality-provenance`.

In fact, in a similar way to what happens in OPM, planning rules are on the one hand related and influenced by the components, and on the other hand they refer to actions in input and they "produce" sets of actions as output.

With respect to the definition of properties, we map the planning predicates to the relations defined in OPM, as follows:

$$\begin{aligned} \text{tpl:hasRuleAction} &\sqsubseteq \text{opmv:used} \\ \text{tpl:isRuleEffectActionOf} &\sqsubseteq \text{opmv:wasGeneratedBy} \\ \text{tpl:hasReferenceComponent} &\sqsubseteq \text{opmv:wasControlledBy} \\ \text{tpl:actionTriggersAction} &\sqsubseteq \text{opmv:wasDerivedFrom} \\ \text{tpl:ruleTriggersRule} &\sqsubseteq \text{opmv:wasTriggeredBy} \end{aligned}$$

3.2 Completion rules and inferences

The definition of the planning OPM profile enables a set of completion rules and inferences [3], i.e. a set of rules that allows to derive further provenance relationships between processes and artifacts in a provenance graph. In our case, applying those rules to our planning OPM profile can help in inferring indirect or implicit causal relationships between actions and planning rules defined in a planning domain theory.

The OPM completion rules can be summarized as follows:

$$\begin{aligned} \text{opmv:wasTriggeredBy} &\equiv \text{opmv:used} \circ \text{opmv:wasGeneratedBy} \\ \text{opmv:wasDerivedFrom} &\sqsupseteq \text{opmv:wasGeneratedBy} \circ \text{opmv:used} \end{aligned}$$

The first line above formalizes the so-called *artifact introduction and elimination* completion rule: a process was triggered by another process if and only if an artifact used by the first process was generated by the second one.

The second line expresses the *process introduction* completion rule: if an artifact was derived from another artifact, there must have been a process that generated the first artifact and used the second one. OPM explicitly states that in general the converse rule (process elimination) does not hold, because without any internal knowledge of the process, it is not possible to assert an actual dependency between the two artifacts. However, OPM also offer the possibility to relax this constraint within a specific OPM profile; in our case, the processes are always planning rules which – by definition – express the causality dependency between actions (i.e., artifacts). Thus, in our planning OPM profile, we state that both the *process introduction* and the *process elimination* completion rules hold, replacing the second line of the axioms above with the following one:

$$\text{opmv:wasDerivedFrom} \equiv \text{opmv:wasGeneratedBy} \circ \text{opmv:used}$$

Thus, because of the mapping we defined above, in our planning OPM profile, the following completion rules hold:

$$\begin{aligned} \text{tpl:ruleTriggersRule} &\equiv \text{tpl:hasRuleAction} \circ \text{tpl:isRuleEffectActionOf} \\ \text{tpl:actionTriggersAction} &\equiv \text{tpl:isRuleEffectActionOf} \circ \text{tpl:hasRuleAction} \end{aligned}$$

Those completion rules let us derive the causal relationships between any couple of actions or processes in a planning domain theory.

Additionally, OPM define *multi-step inferences* to account for indirect causes of an artifact or a process as effect of multiple steps. Specifically, OPM defines the multi-step version of `wasDerivedFrom`, `used`, `wasGeneratedBy` and `wasTriggeredBy` edges (which all depend on the transitive closure of the `wasDerivedFrom` relation). We can express those multi-step inferences as follows:

$$\begin{aligned} & \textit{Transitive}(\,\mathsf{opmv{:}wasDerivedFrom}\,) \\ \mathsf{opmv{:}used} &\sqsubseteq \mathsf{opmv{:}used} \circ \mathsf{opmv{:}wasDerivedFrom} \\ \mathsf{opmv{:}wasGeneratedBy} &\sqsubseteq \mathsf{opmv{:}wasDerivedFrom} \circ \mathsf{opmv{:}wasGeneratedBy} \\ \mathsf{opmv{:}wasTriggeredBy} &\sqsubseteq \mathsf{opmv{:}used} \circ \mathsf{opmv{:}wasDerivedFrom} \\ &\quad \circ \mathsf{opmv{:}wasGeneratedBy} \end{aligned}$$

Thanks to the mapping between our planning metamodel and the OPMV, the following multi-step inference rules hold:

$$\begin{aligned} & \textit{Transitive}(\,\mathsf{tpl{:}actionTriggersAction}\,) \\ \mathsf{tpl{:}hasRuleAction} &\sqsubseteq \mathsf{tpl{:}hasRuleAction} \circ \mathsf{tpl{:}actionTriggersAction} \\ \mathsf{tpl{:}isRuleEffectActionOf} &\sqsubseteq \mathsf{tpl{:}actionTriggersAction} \circ \mathsf{tpl{:}isRuleEffectActionOf} \\ \mathsf{tpl{:}ruleTriggersRule} &\sqsubseteq \mathsf{tpl{:}hasRuleAction} \circ \mathsf{tpl{:}actionTriggersAction} \\ &\quad \circ \mathsf{tpl{:}isRuleEffectActionOf} \end{aligned}$$

Again, those inferences can be employed to derive indirect causal relationships between actions and processes.

Summing up, the definition of an OPM profile for our planning metamodel allows us to reuse the provenance primitives to analyse and infer new knowledge about the causality between actions and planning rules in a domain theory. In the following section, we will show how the above inferences can help in checking the planning domain modelling.

4 Automated Checking of Causality in Planning Models

Whereas planning software is supposed to conform to a solution search algorithm, a planning domain model is supposed to capture a piece of reality and so it requires its own acceptance criteria and tests. Using our ontological formulation of the planning metamodel, we can devise guidelines and tests to capture different levels of consistency for domain models. We would like to stress that, while the typical concerns of a *planner* are efficiency, correctness and completeness of the planning algorithms, here we concentrate on the *modeller*'s point of view, whose concerns include validation, expressive power and maintenance of a domain model [12].

In this section, we give some examples of controls that are enabled by our ontological formulation of the planning metamodel and by its mapping to OPM. Those controls can be successfully employed to support the modeller's task, i.e. to verify the correctness and consistency of the planning domain model. Without claiming to be exhaustive, in the following we formalize some of those controls

and we explain how those checks can be easily implemented via SPARQL 1.1 [10] queries.

4.1 Model Completeness and Action Reachability

As outlined in [12], usually modellers start from the identification of the relevant objects (planning components), then continue with the definition of their relations (actions on components), afterwards they analyse the possible world states and their transitions (planning rules), and so on. When modelling a large or complex planning domain, the number of introduced entities can be very high and verifying the consistency and meaningfulness of the whole model can become complex.

We define a planning domain model as *complete* when all modelled components are involved in some action and all modelled actions are involved in some planning rule. It is worth noting that "orphan" components or actions does not make the domain theory inconsistent per se, but they can suggest an unfinished or lacking modelling. Setting a control to check model completeness is aimed to support the modeller to identify potential lacks or shortcomings in the domain definition.

Component and actions making the model incomplete can be defined as:

$$\begin{aligned} \mathsf{tpl{:}OrphanComponent} &\sqsubseteq \mathsf{tpl{:}Component} \sqcap \forall\, \mathsf{tpl{:}isActionOf}^{-}.\bot \\ \mathsf{tpl{:}OrphanAction} &\sqsubseteq \mathsf{tpl{:}Action} \sqcap \forall\, \mathsf{tpl{:}isRuleEffectActionOf}.\bot \\ &\quad \sqcap \forall\, \mathsf{tpl{:}hasRuleAction}^{-}.\bot \end{aligned}$$

Checking the completeness of the model therefore means that the above defined classes have no instances in the planning domain model. Conversely, if some "orphans" are found, those are the domain entities the modeller should look at to identify potential pitfalls. Assuming a closed world assumption, we can implement this check simply by querying the planning domain model expressed in our planning metamodel with the following SPARQL 1.1 [10] queries:

```
SELECT ?component
WHERE {
  ?component a tpl:Component .
  FILTER NOT EXISTS { ?action tpl:isActionOf ?component . }
}
```

```
SELECT ?action
WHERE {
  ?action a tpl:Action .
  ?rule a tpl:PlanningRule .
  FILTER NOT EXISTS { ?rule tpl:hasRuleAction ?action . }
  FILTER NOT EXISTS { ?action tpl:isRuleEffectActionOf ?rule . }
}
```

Another property of a domain model a modeller could wish to check is *action reachability*. We define an action reachable if there is at least a planning

rule which causes that action, i.e. which has that action as effect. Again, this kind of control is aimed at supporting the modeller to identify potentially incomplete entity definition: if an action is introduced in a domain, it is very likely that the modeller considered it possible to generate that action in some plan. Nevertheless, it also perfectly reasonable that a defined action appears only as the condition to fire a planning rule and not as its effect. This can happen when the action refers to an uncontrollable component (cf. Section 2.1): in this case, since the action activation depends on an agent external to any generated plan, there is no need for a planning rule to cause that action in the domain model.

An unreachable action can be defined as follows:

$$\begin{aligned} \mathsf{tpl{:}UnreachableAction} \sqsubseteq\ & \mathsf{tpl{:}Action} \sqcap \forall\, \mathsf{tpl{:}isRuleEffectActionOf} \,.\, \bot \\ & \sqcap \forall\, \mathsf{tpl{:}isActionOf} \,.\, \mathsf{tpl{:}ControllableComponent} \end{aligned}$$

Following the definition of the planning OPM profile (cf. Section 3.1), the class above can be also expressed as follows:

$$\begin{aligned} \mathsf{tpl{:}UnreachableAction} \sqsubseteq\ & \mathsf{opmv{:}Artifact} \sqcap \forall\, \mathsf{opmv{:}wasGeneratedBy} \,.\, \bot \\ & \sqcap \forall\, \mathsf{opmv{:}used}^{-} \,.\, \mathsf{opmv{:}wasControlledBy} \,.\, \\ & \mathsf{tpl{:}ControllableComponent} \end{aligned}$$

Thanks to OPM multi-step inferences (cf. Section 3.2), this definition includes all possible provenance paths that connect actions (artifacts) with planning rules (processes).

Again, checking the action reachability in a domain model means verifying that the above defined class has no instances in the domain theory. To identify the unreachable actions, and thus understand the appropriateness of their definition, a modeller can use the following SPARQL 1.1 query (or the respective one that makes use of OPMV properties as per the planning OPM profile):

```
SELECT ?action
WHERE {
  ?action a tpl:Action.
  FILTER NOT EXISTS {
    ?rule tpl:hasRuleEffect/tpl:hasActionValue ?action .
  }
  FILTER NOT EXISTS {
    ?action tpl:isActionOf [ a tpl:UncontrollableComponent ] .
  }
}
```

4.2 Constraint Checking

As illustrated in Section 2, defining a planning rule includes also the introduction of a set of conditions, i.e. constraints on the rule activation. It is often the case that most of the "rational" and complexity of a planning domain theory lies in its rules' conditions. Therefore, verifying the consistency of a domain model means checking the satisfiability of the constraints defined in the planning rules.

For example, let's say that we want to capture the rules of an educational institution. A planning rule could say that, when a new student arrives and asks to join the school, if he/she is above legal age (condition), the school can enrol him/her. Another planning rule could state that, if a student is hurt and he/she is below legal age (condition), the school should inform his/her parents. The previous two rules are both reasonable and the first one is a sort of pre-condition for the second one (people are considered students only after their enrolment); still, it is apparent that the second rule will never be triggered, since no student of this institution can be under legal age. Thus, a modeller has to check all rules' conditions to understand if they ever apply.

In our planning metamodel, we decided to assimilate rule conditions to SPARQL FILTER and LET clauses (cf. Section 2.3). This modelling choice comes of help also for constraint checking: starting from the condition definition, it is natural to create SPARQL queries with those clauses, in case combining different conditions; executing those queries on the planning domain model helps the modeller to identify inconsistencies and potential modelling mistakes.

5 Applying our approach to Simulation Learning

Simulation Learning is a kind of training aimed to improve soft skills [13]. Simulation Learning systems generally re-create near-real environments for training sessions, in which learners are subject to stimuli: they have to learn how to deal with the simulated situation and how to react to it. Such simulations need to be effective and engaging, so that the learners do not simply memorise notions, but they actively and permanently acquire skills, practice and knowledge. In this context, simulation sessions can be generated using planning technology from a learning domain model.

The Pandora project[4] aims to provide a platform for Crisis Management simulation learning, providing a near-real training environment at affordable cost. The Pandora platform [14] makes use of Timeline-based Planning technologies [7, 8] to plan the simulation sessions and it exploits the ontological framework explained in this paper to represent the actions causality in the crisis simulation scenario. To this end, we specialized the planning ontology introduced in Section 2.2 with the relevant entities of Crisis Management training, thus specifying the Pandora ontology[5] and we set up a Linked Data-empowered Knowledge Base [15] to manage the domain theory definition.

Modelling a Crisis Management scenario – or any Simulation Learning scenario – means creating the crisis events to stimulate trainees (e.g., a street is flooded, a hospital electricity becomes scarce for a black-out) and to plan for different storyboard evolutions in response to trainees' actions (e.g., depending on the Crisis Managers decisions, the crisis situation becomes more or less critical).

We applied the approach outlined in Section 4 to support the modeller in verifying the domain theory: while not substituting the manual intervention,

[4] Cf. `http://www.pandoraproject.eu/`.
[5] Cf. `http://swa.cefriel.it/ontologies/pandora`.

this method proved to be a useful means to detect potential problems, before checking the plans generated by the planning algorithms. Indeed, the adoption of our ontological metamodel and its verification via SPARQL queries (both the generic completeness and reachability controls from Section 4.1 and some manually-defined and domain-dependent queries built on top of the Pandora rules conditions as described in Section 2.3) allowed the modeller to identify missing definitions and unreachable actions. Though preliminary and qualitative, this evaluation makes us believe in the usefulness and efficacy of our proposed approach. We are currently investigating an automated way to generate the SPARQL queries needed for constraint checking, starting from the planning rule conditions' definition.

6 Related Work

Different works in literature dealt with the ontological representation of planning and scheduling knowledge. Early models like [16] and [17] were aimed at representing planning tasks and planning problems; however, they were limited in scope and did not considered all the relevant entities of the planning world. The most comprehensive planning ontology so far is described in [18]: it is the first formalization that includes the temporal dimension and the notion of agents. Still, that model was aimed at proposing an operational specification of the planning problem with a set of "executable" definitions.

In contrast, our planning metamodel is not directly oriented to the search for plans; with our formalization we aim at supporting the modelling of planning domain theories, by identifying potential problems prior to the execution of the planning algorithms. Moreover, our metamodel is more detailed than the ontology described in [18], because we make the constraints on the planning rules "first-class citizens" of our conceptualization. This `Condition` concept not only allows for the declarative definition of a complete domain theory, but it also provides a mechanism to reuse and share constraints between different planning rules. Furthermore, we underscored the temporal aspect of planning by introducing the `TemporalCondition` primitive and by modelling the `TemporalFunctions` hierarchy.

In the planning community, a dedicated workshop [19] was organized to discuss "the role of ontologies in Planning and Scheduling". The result was that ontological languages like RDFS and OWL are more expressive than the ones in the planning field, for example because of the Open World Assumption; however, they cannot be "directly applied" in planning systems because they are usually employed to represent static knowledge. While we agree with this statement, our investigation is oriented precisely in the possible cooperation between ontologies and planning on their "boundary": we can say that our metamodel statically captures the dynamics of planning. Thus we are convinced that ontology-based knowledge representation can bring benefits to current planning technologies [20].

Finally, a note about the languages used in planning: PDDL [2] is currently the most popular, even if its iterative standardization did not prevent the spread of a multitude of different dialects. While a comparison of expressivity between PDDL and ontological languages is out of scope of this paper[6], we would like to stress again that we propose the use of an ontological formalization *outside* the search process to find a solution to the planning problem. Our formalization and causality checks are complementary to the consistency and validation of the plans generated in the solution space.

7 Conclusions

In this paper, we presented our approach to formalize the planning primitives and their relations with an ontology. We believe that this is a good example of interplay between Semantic Web and Planning technologies [20], since knowledge representation and reasoning – even in simple forms as we did in this work – can be of great help during the planning modelling: the automatic checking of the domain theory characteristics (e.g., completeness and reachability as defined in this paper) supports the modellers' job, because it helps them to identify potential problems in their modelling even before checking the consistency of the generated plans. Moreover, because of the spread and success of knowledge sharing on the Web, including the Linked Data movement, planning modelling can be simplified or reduced by reusing and linking to pre-existing datasets, as we illustrated in a previous work [15].

To this end, we introduced an OWL 2 representation of this planning metamodel and its formulation as an Open Provenance Model Profile, through a mapping between our metamodel and the OPM Vocabulary. The reason why we chose to adopt OPM is two-fold. On the one hand, the *provenance* abstractions are very similar to the ones used in *causality* models, like those we have in our planning metamodel. On the other hand, the completion rules and inferences defined in OPM offer a simple yet powerful means to perform checks; this is why we leveraged those reasoning means to formulate our model checks.

Our future works are oriented in two directions: extending the causality checks introduced in this paper and applying analysis and mining on the actual "executions" of plans generated on the basis of domain theories represented in our metamodel. With regards to the former, we will take into account some more characteristics already modeller in the planning ontology introduced in Section 2 and enhance accordingly the causality controls defined in Section 4.

Regarding the latter, once a domain theory has been modelled and used to generate plans, those plans can be "executed" in planning applications (e.g., in the simulation learning scenario introduced in Section 5, when training sessions take place); the recording of those executions can be seen as "streams" of events, i.e. assertions that are valid in a specific time-frame. We believe that we can exploit those time-stamped assertions to refine the causality modelling: e.g.,

[6] A discussion of the possible interplay between PDDL and Datalog is offered in [21].

comparing the actual streams of happened events in different sessions, we can identify the most frequent patterns of events which have some causality aspect as well as the least frequent plan options; those can be interesting hints for the planning modeller to improve the domain theory. Dealing with plan executions means taking in consideration the *temporal* dimension, thus we need proper approaches; to this end, we aim at employing Stream Reasoning technologies [22], also by following other experiences in ex-post provenance analysis of workflow executions as in [23].

Acknowledgments

This research is partially funded by the EU PANDORA project (FP7-ICT-2007-1-225387). We would like to thank the project partner for their collaboration.

References

1. Fikes, R., Nilsson, N.: STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence **2** (1971) 189–208
2. Gerevini, A., Long, D.: Plan Constraints and Preferences in PDDL3. Technical report, R.T. 2005-08-47, Dipartimento di Elettronica per l'Automazione, Università degli Studi di Brescia (2005)
3. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., den Bussche, J.V.: The Open Provenance Model core specification (v1.1). Future Generation Computer Systems (2010)
4. Zhao, J.: Open Provenance Model Vocabulary Specification. Online at `http://purl.org/net/opmv/ns` (2010)
5. Hitzler, P., Kroetzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 Web Ontology Language Primer. W3C Recommendation, `http://www.w3.org/TR/owl2-primer/` (October 27, 2009)
6. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education Inc. (2003)
7. Cesta, A., Fratini, S.: The timeline representation framework as a planning and scheduling software development environment. In: 27th Workshop of the UK Planning and Scheduling SIG. (2008)
8. Cesta, A., Cortellessa, G., Fratini, S., Oddi, A.: Developing an end-to-end planning application from a timeline representation framework. In: 21st Applications of Artificial Intelligence Conference. (2009)
9. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11) (1983) 832–843
10. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. W3C Working Draft, available at `http://www.w3.org/TR/sparql11-query/` (2011)
11. Knublauch, H.: SPIN Modeling Vocabulary. `http://spinrdf.org/spin.html` (October 20, 2009)
12. Mccluskey, T.L., Porteous, J.: Engineering and Compiling Planning Domain Models to Promote Validity and Efficiency. Artificial Intelligence **95** (2000) 1–65
13. Aldrich, C.: Simulations and the Future of Learning: An Innovative (and Perhaps Revolutionary) Approach to e-Learning. Pfeiffer (2003)

14. Bernardi, G., Cesta, A., Coraci, L., Cortellessa, G., De Benedictis, R., Mohier, F., Polutnik, J., Vuk, M.: Only Hope remains in the PANDORA's .jar – Pervasive use of planning in a training environment. In: 21st International Conference on Automated Planning and Scheduling, System Demonstrations and Exhibits, Best Demo Award. (2011)
15. Celino, I., Dell'Aglio, D.: A Linked Knowledge Base for Simulation Learning. In: Proceedings of the 1st International Workshop on eLearning Approaches for the Linked Data Age (Linked Learning 2011), co-located with the 8th Extended Semantic Web Conference, ESWC2011. (2011)
16. Mizoguchi, R., Vanwelkenhuysen, J., Ikeda, M.: Task Ontology for Reuse of Problem Solving Knowledge. In: Towards Very Large Knowledge Bases. IOS Press (1995) 46–57
17. Gil, Y., Blythe, J.: Planet: A sharable and reusable ontology for representing plans. In: the AAAI - Workshop on Representational Issues for Real-World Planning Systems. (2000) 28–33
18. Rajpathak, D., Motta, E.: An ontological formalization of the planning task. In: International Conference on Formal Ontology in Information Systems (FOIS'04). (2004) 305–316
19. Olivares, J.F., Onaindia, E., eds.: Workshop on the Role of Ontologies in Planning and Scheduling, co-located with the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005). (2005)
20. Celino, I., Dell'Aglio, D., De Benedictis, R., Grilli, S., Cesta, A.: Ontologies, rules and linked data to support crisis managers training. IEEE Learning Technology Newsletter, Special Issue "Semantic Web Technologies for Technology Enhanced Learning" **13** (2011) Issue 1
21. Thiebaux, S., Hoffmann, J., Nebel, B.: In Defense of PDDL Axioms. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003). (2003) 961–968
22. Della Valle, E., Ceri, S., van Harmelen, F., Fensel, D.: It's a Streaming World! Reasoning upon Rapidly Changing Information. IEEE Intelligent Systems **24**(6) (2009) 83–89
23. Miles, S., Wong, S.C., Feng, W., Groth, P., Zauner, K.P., Moreau, L.: Provenance-based validation of e-science experiments. Journal of Web Semantics **5**(1) (2007) 28–38