Verifying the compliance of personalized curricula to curricula models in the semantic web*

Matteo Baldoni, Cristina Baroglio, Alberto Martelli
Viviana Patti, and Laura Torasso

Dipartimento di Informatica — Università degli Studi di Torino
C.so Svizzera, 185 — I-10149 Torino (Italy)
{baldoni,baroglio,mrt,patti,torassol}@di.unito.it

Abstract. In this work we propose the introduction of a decoupling between personalized curricula and curricula models. A curricula model is formalized as a set of time constraints, while personalized curricula are formalized by means of an action theory. Given this framework, it is possible to make various interesting verification tasks automatic. In particular, we will discuss the possibility of verifying the compliance of personalized curricula to models, by using temporal reasoning. Compliance verification allows to check the soundness of a curriculum customized w.r.t. available resources and user goals against a model that expresses temporal learning dependencies at the knowledge level.

1 Introduction

The Semantic Web is concerned with adding a semantic layer to resources that are accessible over the internet in order to enable sophisticated forms of re-use and reasoning. In the last years standard models, languages, and tools for dealing with machine-interpretable semantic descriptions of Web resources have been developed. In this context a strong new impulse to research on personalization can be given: the introduction of machine-processable semantics makes the use of a variety of reasoning techniques for implementing personalization functionalities possible, widening the range of the forms that personalization can assume.

Learning resources are particular kind of resources specifically useful in an educational framework. Especially with the development of peer-2-peer and service oriented e-learning architectures, it become fundamental to explore solutions for personalizing w.r.t. the user's needs the retrieval and the composition of learning web resources. In our opinion sophisticated personalization functionalities

* This research has partially been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. http://rewerse.net), and it has also been supported by MIUR PRIN 2005 "Specification and verification of agent interaction protocols" national project.

should combine lesson learnt in the community of traditional educational systems (especially for what concerns the re-use of learning resources), and the new possibility of running reasoning techniques developed in the AI community over the semantically annotated learning resources.

In recent years, the *educational systems* community has focussed greater and greater attention to the problem of separating the *contents* of learning resources, from the *means* that is necessary for taking advantage of the contents. The chief goal is to enable a reuse of the learning resources, where re-use is more and more often intended as a process by which the contents of a new complex learning resource, e.g. a course, are assembled, at least partly, starting from already encoded contents, the optimal situation being a complete decoupling of the resources from the platforms used for playing them. A first significant step in this direction is represented by the birth of SCORM [1] and of Learning Design [13, 14]. The former allows to build a new course (formally, a new SCO) on top of existing SCOs or assets. The latter, is focussed on the design of processes and workflows among a group of actors that take part to the learning activities. These tools, however, suffer the lack of a machine-interpretable information about the learning resources, for enabling forms of automatic composition and of verification, possibly based on reasoning.

Standard languages for semantic annotation like RDF [16] and LOM [12] can be used for filling this lack and adding some meta-data to the resources. In particular by meta-data we can supply information on the learning resources at the *knowledge level*, e.g. knowledge about the *learning objectives* of the resource and its *prerequisites*. Given such kind of annotation, we can interpret a learning resource as an action, that can profitably be used if the learner has a given set of competences (preconditions); by using it, the learner will acquire a new set of competences (effects). As we have shown in previous work [3, 2], given an annotation of resources with preconditions and effects one can rely on a classical theory of actions and applying different reasoning techniques for offering different kind of personalization functionalities. For instance, one could use *classical planning* for performing curriculum sequencing, i.e. for selecting and sequencing a set of resources which will allow a user to achieve her/his learning goal [3]. Moreover it possible to exploit *temporal projection* for validate a student-given curriculum verifying whether all the preconditions are respected [2]. Last but not least it is possible to exploit *procedural planning* for performing curriculum sequencing at the level of university courses, in order to help a student to customize a curriculum offered by the University w.r.t his/her interest [2]. In our previous work all these tasks were accomplished by exploiting the metaphor of learning resources as actions and the representation at the knowledge level of the student learning goal and knowledge profile. We exploited a reasoning engine based of the logic language DyLOG [4], that provided a unified framework for performing both classical and procedural planning and temporal projection.

In this work we aim at taking a step further on this line of research and we focus on a kind of verification that can be profitably combined with the curriculum sequencing personalization functionalities investigated in previous

work, by leading to implement sophisticated personalization applications in a unified framework. Given a semantic annotation of the resources based on the metaphor of resources as actions, we will focus on a new kind of reasoning, which can be accounted as a *compliance* verification of personalized curricula w.r.t. a curricula model. Personalized curricula are intended as learning paths through learning resources personalized w.r.t. specific user need, e.g. they could be the result of a curriculum sequencing method that exploits the planning techniques mentioned above. Curricula models specify general rules for building such paths and can be interpreted as constraints. These constraints are to be expressed in terms of knowledge elements, and maybe also on features that characterize the resources. If such resources are courses, they should not be based, generally speaking, on specific course names. So a constraint might impose a lab course to be attended after a theory course on the same topics but not that the course C123 should follow C122.

Verifying the compliance a curriculum to a model means checking: first of all, that the resources are sequenced in such a way that their preconditions are respected, that the learning goal is achieved in the end, and that along the sequence the constraints imposed by the model are satisfied. In the following we present a preliminary proposal for a knowledge representation that suits the outlined problem domain and sketch the techniques by which the comparison of courses to constraint-based schemas can be performed.

Compliance verification can be useful in many practical cases where the need of personalizing learning resource sequencing w.r.t. to the student desire has to be *combined* with the ability to check that the result of personalization fit some abstract constraints, possibly imposed by a third party. A given University could, for instance, certify that the specific curricula that it offers for achieving a certain educational goal -that built upon the local university courses- respect some European schemes defined at the abstract level of competence. Such automatic checking of compliance combined with curriculum sequencing techniques could be used for implementing processes like cooperation in curriculum design and curricula integration which are actually the focus of the so called *Bologna Process* [8], promoted by the EU ministers responsible for higher education: "Curriculum design means drawing up of a common study path aimed at reaching the educational goals that have been jointly defined. In these schemes the partners offer specific segments which complement the overall curriculum designed". Further use cases are sketched in the conclusions.

2 Knowledge representation and verification

In this section we discuss about the possible formal representations of *specific curricula*, intended as sequences of learning resources (be they documents or other forms of learning materials or, at a higher level, entire courses) and *curricula models*, intended as specifications of general schemata for achieving a certain *educational goal*, where relationships among competencies are described.

2.1 Description of resources based on an action theory

Let us consider a specific curriculum as a sequence of resources, that are homogeneous in their representation. Based on work done in previous research [2, 3], we can represent such resources in an action theory, taking the abstraction of resources as simple actions. The idea is to introduce at the level of the learning resources, some semantic annotation that describe both their pre-requisites and effects, as done in the curriculum sequencing application in [3]. Regarding the semantic annotation, it could be done by LOM, that allows the annotation of the learning objects by means of an ontology of interest by using the attribute classification as suggested in [3].

Let us summarize our intuition of resources as actions. An action can be executed given that a set of pre-conditions holds; by executing it, a set of post-conditions -called also effects- will become true.

In our domain of application we can interpret a learning resource as the action of acquiring some knowledge elements (effects). According to this metaphor, a resource can be used only if the user owns given knowledge elements or competencies (preconditions). Thus, a resource can be described in terms of knowledge elements. For instance let us use a classical STRIPS-like notation for describing the resource called *db_for_biotech* with prerequisites *relational databases* and effects *scientific databases* as:

ACTION: db_for_biothec(),
PREREQ: relational_db, EFFECTS: scientific_db

A curriculum is a sequence of actions of the kind reported above.

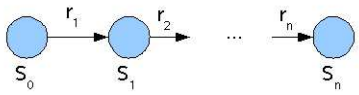

Fig. 1. The labels on the edges, r_1, r_2, ... r_n, represent learning resources. The states S_i represent sets of competences that are available at a given time.

Actions in the sequence cause transitions from a state to another, starting from an initial state up to a final state. The *initial state* represents the initial set of competences that we suppose available before the curriculum is taken (e.g. the basic knowledge that the student already has). This set can also be empty. The subsequent states are obtained by applying the actions (resources) that tag the transitions. Each of such actions has a set of preconditions that must hold in the state to which the action is applied and cause some modifications that consist in an update of the state. The prerequisites of action r_i must hold in the state S_{i-1}. The state S_i is obtained by adding to S_{i-1} the effects of r_i. See Figure 1. We assume that competences can only be added to a state

after executing the action of attending a course (or more in general reading a learning material). The intuition behind this assumption is that no new course will ever erase from the students memory the concepts acquired in previous courses, thus knowledge grows incrementally. Formally, it correspond to assume that the domain is monotonic.

2.2 Curricula models

Curricula models are to be defined on the basis of knowledge elements as well. In particular, we would like to restrict the set of possible sequences of resources, by imposing constraints on the order by which knowledge elements are added to the states. For instance, that a knowledge element α is to be acquired before a knowledge element β or that a knowledge element α is guaranteed to be acquired, or that the acquisition of α implies that also β will be acquired subsequently. Therefore we will represent a curriculum model as a set of *temporal constraints*. Being defined on knowledge elements, a curriculum model is independent from the specific resources that are taken into account, then, it can be used in an open and dynamic world like the web. A set of similar constraints defines a schema that can be used for checking user specific curricula intended as sequences of actual resources.

A natural choice for representing temporal constraints on action paths is linear-time temporal logic (LTL) [7]. This kind of logic allows the verification that a property of interest is true for all the possible executions of a model, which in our case corresponds to the specific curriculum. This is often done by means of model checking techniques [6]. ***INSERIRE QUI MOTIVAZIONE sull'uso di LTL**** The curriculum that we mean to check is, indeed, a Kripke structure; as thus, it is easy to verify properties expressed as temporal logic formulas. Briefly, a Kripke structure identifies a set of states and transition relation that allows passing from a state to another (see Figure 1). In our case, the states correspond to the competencies that are owned at a certain moment. Since we assume the domain is monotonic in the sense pointed out in the previous subsection, states will contain *all* the competencies acquired up to that moment. The transition relation is given by the actions that are contained in the curriculum that is being checked. Since the sequence is linear and shows no branch, then, it is possible to reason on the states and with LTL logic it is possible to verify that a given formula holds starting from a state or that it holds for a set of states.

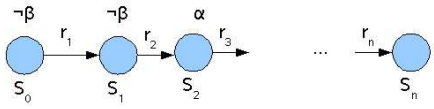

Fig. 2. β can hold only after α becomes true, therefore, in the states that follow S_3 β can either hold or not hold.

For example, the fact that a knowledge element β cannot be acquired before the knowledge element α is acquired, can be written as the LTL temporal formula $\neg\beta \ U \ \alpha$, where U is the *weak until* operator (see Figure 2). Given a set of knowledge elements to be acquired, such constraints specify a partial ordering of the same elements. Other kinds of constraints might be taken into account. For instance, that a knowledge element will be acquired sooner or later ($\Diamond\alpha$, eventually operator). A curriculum model is meant to allow the achievement of

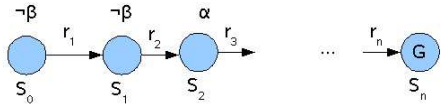

Fig. 3. A curriculum that allows the acquisition of the learning goal $\mathcal{G}$.

a given *learning goal*, that consists in a set of knowledge elements. We expect that the learning goal will hold in the final state of every curriculum that matches with the model (see Figure 3).

2.3 Compliance Verification

Given a representation of a user specific curriculum as sequence of actions/resources $(r_1, r_2, ..., r_n)$ with preconditions and effects, based on knowledge elements in an action theory $(\mathcal{A})$, and a representation of a curricula model, based on temporal constraints $(\mathcal{T})$ and a learning goal $(\mathcal{G})$, it is possible to apply different reasoning techniques for performing various interesting tasks. Besides planning, that we have already explored in previous works [2, 3], in this formal framework we can verify the *compliance* of s user specific curriculum to a model. The verification could be based on temporal reasoning techniques, like temporal projection, and on model checking techniques. Verifying the compliance means, in simple words, to check whether the curriculum respects the model, i.e that the sequence $r_1, ..., r_n$ is sound w.r.t. the precondition and effect relations specified in $\mathcal{A}$, that the sequence allows reaching the goal $\mathcal{G}$, and that the sequence respects the temporal constraints in $\mathcal{T}$. Intuitively, we can think of combining temporal projection and model checking by verifying

$$\mathcal{A} \models_{AL} \mathcal{G} \textbf{ after } r_1, ..., r_n \quad (*)$$

where AL is any action logic that supports temporal projection, and

$$r_1, ..., r_n \models_{LTL} \mathcal{T} \quad (**)$$

where LTL is a linear-time temporal logic.

3 Possible implementation

In the following we discuss the possibility of exploiting existing technology and languages for developing a system that can perform the forms of verifications described above. In particular we will deal both with the selection of languages for the representation of models and of curricula, and with the exploitation of existing tools for performing the verifications.

A semantic representation of an action is quite simple and mostly consists of two lists of knowledge elements: those required for using the resource and those supplied by the resource. In order for the knowledge elements themselves to have a semantic value, they can be implemented as terms in shared vocabulary (the simplest form of ontology). RDF can be used as an implementation language. Resources are used to define curricula. In this work we focus on curricula obtained by sequencing resources, therefore we represent a curriculum as an action sequence. This sequence can be considered as a simple kind of program that contains no branch or loop or recursive call. For what concerns action representation, there is a wide choice of action languages that are valuable candidates: we could use a logic programming action language, like $\mathcal{A}$ by Gelfond and Lifschitz [9], DyLOG [4], or GOLOG [15], all of which provide proof procedures that support temporal projection (*).

Given that a curriculum has passed the temporal projection test, we can use a model checker to verify the temporal constraints (**). Model checking is the algorithmic verification of the fact that a finite state system complies to its specification. In our case the specification is given by the curriculum model and consists of a set of temporal constraints, while the finite state system is the curriculum to be verified. Among the various model checkers that have been developed, it is worthwhile to mention SPIN [11] and NuSMV [5]. SPIN, in particular, is used for verifying systems that can be represented by finite state structures, where the specification is given in an LTL logic. The verification algorithm is based on the exploration of the state space. This is exactly what we need for performing the second step of our compliance test, provided that we can translate the curriculum in the internal representation used by the model checker. In the case of SPIN, the internal representation is given in the Promela language. For example, we can represent the knowledge elements as boolean variables, therefore actions as transitions that modify the values of some of these variables. The constraints will be temporal formulas that use such variables. The verification that the constraint should along the whole curriculum is performed automatically by the model checker.

In the case of linear curricula it would be easy to integrate in the temporal projection algorithm the direct verification of the constraints. The opposite solution of integrating the temporal projection into a model checker, which is the one that we mean to pursue, has the advantage of allowing the extension of the compliance test to curricula that have a more complex structure. In fact, curricula might contain tests, branching points, and repetitions. For example, if the curriculum corresponds to a learning resource that has been assembled on the basis of other learning resources (for instance a SCORM object), it might

contain, as well as a program, also loops. As well as in the two-steps solution described above, it would be necessary to have a translation mechanism that allows turning the representation of the action theory into the internal formalism, used by the model checker [10].

For the sake of completeness, hereafter, we report a part of the Promela code for an example. The code allows the execution of both temporal projection and model checking. Temporal projection is handled as a deadlock verification: if the sequence is correct w.r.t. the action theory, no deadlock arises, otherwise a deadlock will be detected. The complete example and an explanation of it are available at `http://www.di.unito.it/~alice/ccompliance/`. This curriculum passes the compliance test under the temporal constraints $\neg f7\ U\ f5$ and $\neg f8\ U\ f5$. In the web site it is possible to retrieve alse examples of curricula which fail the test.

```
mtype = { course1, course2, course3, course4, course5 };
mtype = { done, stop, success, fail }
chan attend = [0] of { mtype };
chan feedback = [0] of { mtype };
bool f1, f2, f3, f4, f5, f6, f7, f8;

init {    f1 = true; f2 = false; f3 = false; f4 = false;
          f5 = false; f6 = false; f7 = false; f8 = false;
          run TestCompliance();
          run UpdateState();  }

inline Curriculum4() {
    attend!course1; feedback?done;
    attend!course2; feedback?done;
    attend!course5; feedback?done;
}

proctype TestCompliance() {
    Curriculum4()
    feedback!stop; feedback?success;
}

proctype UpdateState() {
    do
    :: attend?course1 -> if
       :: (f1) -> f2 = true; f3 = true; f4 = true; feedback!done;
       fi;
    :: attend?course2 -> if
       :: (f3) -> f4 = true; f5 = true; feedback!done;
       fi;
    :: attend?course3 -> if
       :: (f2 && f6) -> f7 = true; f8 = true; feedback!done;
       fi;
    :: attend?course4 -> if
       :: (f2 && f5) -> f7 = true; feedback!done;
       fi;
```

```
:: attend?course5 -> if
   :: (f2 && f4) -> f7 = true; f8 = true; feedback!done;
   fi;
:: feedback?stop -> if
   :: (f4 && f5 && f8) -> feedback!success;
   :: else -> feedback!fail;
   fi;
   break;
od  }
```

The above program is hand-coded but, as the modularity of the example witnesses, it would be easy to produce an automatic translator able to turn the description of sets of courses and the description of sequences of resources into Promela code. Such code could, then, be validated according to curricula models encoded as sets of temporal constraints.

4 Conclusions

In this work we have presented a two-level representation of curricula, aimed at capturing the distinction between curricula and models of curricula that define general rules or constraints to be satisfied. We have shown that by implementing curricula models as temporal constraints, and curricula as sequences of actions, it is possible to verify the compliance of a curriculum to a model by exploiting reasoning techniques that combine temporal projection and model checking.

The possibility of verifying the compliance of curricula to models is extremely important in many applicative contexts where the need of personalizing learning resource sequencing w.r.t. to the student desire has to be combined with the ability to check that the result of personalization fit some abstract models. In this sense we can say that the compliance verification we propose is complementary w.r.t the capability of applying planning techniques for building from a set of available resources, personalized curricula aimed at reaching a given learning goal. Representing models as sets of constraints gives great freedom in the definition of specific curricula because it cuts away the undesired curricula without imposing unnecessary constraints. The same freedom is not supplied if we represent, as in [2], models as procedures. Procedures have a prescriptive nature that over-rules the possible solutions; the greater flexibility introduced by the use of temporal constraints has a positive effect on the possible personalization of the solutions, by allowing a greater autonomy in selecting among alternatives.

Concerning use cases, we have already mentioned the Bologna process. Another practical application could be helping a teacher that must teach a same topic to different classes, with background and purposes that vary. For instance, to teach Java to a University class as well as to professionals that work in an information technology enterprise. The teacher might be interested in the fact that all students of both classes acquire a same set of competences, with known time constraints, however, since the target students are so different it is useful to prepare two different courses exploiting different learning resources. The University students must, in fact, be taught also the theoretical background concerning

object-oriented programming. On the other hand, the professionals will surely be more interested in more practical lessons, containing many real-world examples of application. The teacher might select public-domain (semantically annotated) learning resources from on-line repositories and use them to compose two different curricula personalized w.r.t. the different student targets. Nevertheless, by applying the approach that we have proposed he would have the possibility of verifying that the built curricula respect an abstract curriculum schema, derived from the expertise and the experience of the teacher himself.

We are working at the actual development of a system on the line of the sketch described in the previous section. Moreover, we are thinking to an extension (both from a formal and an implementation perspective), in which hierarchies of knowledge elements are used instead of plain vocabularies. Hierarchies allow a representation of knowledge elements at different levels of abstraction, thus they would allow other forms of verification. In order to include them, it might be necessary to integrate forms of ontological reasoning in the framework.

Acknowledgements

The authors would like to thank prof. Nicola Henze for the precious discussions.

References

1. ADLnet. Scorm specification. Available at http://www.adlnet.org.
2. M. Baldoni, C. Baroglio, and V. Patti. Web-based adaptive tutoring: an approach based on logic agents and reasoning about actions. *Artificial Intelligence Review*, 2004.
3. M. Baldoni, C. Baroglio, V. Patti, and L. Torasso. Reasoning about learning object metadata for adapting scorm courseware. In In L. Aroyo and editors C. Tasso, editors, *Proc. of EAW'04: Methods and Technologies for personalization and Adaptation in the Semantic Web*, pages 4–13, 2004.
4. M. Baldoni, L. Giordano, A. Martelli, and V. Patti. Programming rational agents in a modal action logic. *Annals of Mathematics and Artificial Intelligence*, 2004.
5. Alessandro Cimatti, Edmund M. Clarke, Fausto Giunchiglia, and Marco Roveri. NUSMV: A new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.
6. O. E. M. Clarke and D. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 2001.
7. E. A. Emerson. Temporal and model logic. In *Handbook of Theoretical Computer Science*, volume B, pages 997–1072. Elsevier, 1990.
8. Education European Commission and Training. The bologna process. http://europa.eu.int/comm/education/policies/educ/bologna/bologna_en.html.
9. M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–321, 1993.
10. Fausto Giunchiglia and Paolo Traverso. Planning as model checking. In *ECP*, pages 1–20, 1999.
11. Gerard J. Holzmann. The model checker SPIN. *Software Engineering*, 23(5):279–295, 1997.

12. Learning technology standards committee. draft standard for learning object metadata, 2002.
13. IMSGlobal. Learning design specifications. Available at http://www.imsglobal.org/learningdesign/.
14. R. Koper and C. Tattersall. *Learning Design: A Handbook on Modelling and Delivering Networked Education and Training*. Springer Verlag, 2005.
15. H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *J. of Logic Programming*, 31:59–83, 1997.
16. W3C. RDF Primer. Available at http://www.w3.org/TR/rdf-primer/.