Model Checking Syllabi and Student Careers *

Roberto Sebastiani[1], Alessandro Tomasi[1], and Fausto Giunchiglia[1,2]

[1] University of Trento, I-38050 Povo, Trento, Italy,
[2] ITC-IRST, I-38050 Povo, Trento, Italy
{rseba,atomasi,fausto}@cs.unitn.it

Abstract. Model checking has been conceived as a powerful tool for hardware, software and protocol verification, which has its main application fields in the development of hi-tech and safety-critical systems. We present here a completely novel application in the field of university administration processes, in which model checking is applied to the verification of the coherence of syllabi and to the automated synthesis/simulation of correct student careers under given requirements.

1 Motivations and goals

Recently the Italian Ministry of University and Scientific & Technological Research, MURST —*the Ministry* from now on— has approved the reform of the Italian university, which will come into force in the year 2001-2002 [MURST 2000a]. The result is a complex hybrid between the Anglo-Saxon and the traditional Italian university organizations. In order to forestall the many problems arising from the reform, the University of Trento and CINECA [1] have started a joint project, named SS2, whose goal is to build a model of the post-reform structure and administration processes of a university, and to develop a new information system for the university admission & examination offices.

One major problem with the reform evidenced by the project model is the much increased difficulty of reasoning on regulations, syllabi and student careers. For instance, the new autonomy given to the universities in organizing their degrees allows them to drastically enlarge the number of degrees of freedom for the student's career choices. Moreover, the introduction of new distinct didactic activity types, and of the different weights in credits, has further complicated both the planning and the verification of the student careers.

To partially cope with this problem, the project model allows for encoding the ordinance regulations for student careers as a set of *formal rules*, so that they can be read and understood by an automated device. This allows for an automatic verification of the correctness of a career wrt. the rules.

---

* The first and the third author are part of the SS2 project team. The other team members provided feedback on the formalization. Alessandro Cimatti, Marco Roveri and Paolo Traverso from ITC-IRST provided help on Symbolic Model Checking.

[1] Italian university consortium for automated computing.

In this paper we go much further, and we describe how to apply model checking to solve the much harder problems of automatically planning/synthesizing correct student careers under given requirements, and of automatically verifying the coherence of syllabi wrt. the rules. The ultimate goal is to develop an automated support tool, to be integrated with the project's information system.

The paper is organized as follows. In Section 2, we describe in detail the domain and the problems of both career synthesis and syllabus coherence verification; in Section 3, we show how we encode the latter two problems as CTL model checking problems; in Section 4, we describe a prototype tool we have implemented; in Section 5, we present and discuss some preliminary empirical results; in Section 6, we discuss the ongoing and future work.

As a notational remark, many of the Italian terms we use do not have a straightforward translation into English. This is due to the fact that there is not a direct correspondence between the structure and administration processes of Italian and Anglo-Saxon universities. (E.g., the meaning of *ordinance* here may not be the same as it is in Oxford or Harvard.) Thus we will define the meaning of all non-obvious terms explicitly, reporting the corresponding Italian terms.

2 The problem

2.1 A model for the domain

Didactic activities and syllabi By *didactic activity* ("attività didattica") we mean any activity a student can perform to enhance his/her career. Examples of didactic activities are, courses, seminars, theses, stages, projects. Most didactic activities are courses. Subjects are grouped into *subject area*s ("settore scientifico-disciplinare"), whose complete list is published by the Ministry [MURST2000b]. The estimated workload related to each activity is measured in *credits*. At the beginning of the year, the student has to register to a new *matriculation year* ("anno di corso"), which is either the successor of the old one —if the student is reasonably on schedule— or the old one itself —if the student is behind schedule and decides, or is forced, to repeat the year.

A didactic activity is modeled as a record; the fields which are relevant for our discussion are the following:

- the activity *code*;
- one activity *type*. Possible values are *course*, *seminars*, *stage*, *thesis*, etc.;
- one *subject area*;
- one *weight in credits*;
- the list of the codes of the *prerequisite didactic activities*; [2]
- the minimum *matriculation year* at which one can perform the activity.

Some examples of courses are given in Figure 1.

[2] Prerequisites obtained by transitivity are omitted.

	Code	Type	Subject area	Credit weight	Prerequisite courses	Matriculation year
1.	CSI	$course$	$INF01$	5	$\{\}$	1
2.	$CSII$	$course$	$INF01$	5	$\{CSI\}$	2
3.	OOP	$course$	$INF01$	5	$\{CSII\}$	2
4.	GEO	$course$	$MAT03$	5	$\{\}$	1
5.	AI	$course$	$INF01$	5	$\{CSI\}$	3
	...	...	...	...	...	...

Fig. 1. Examples of Courses.

Every year, for every degree course ordinance, the proper entity (faculty, department, ...) presents a *syllabus* ("offerta didattica"), i.e., the list of courses which are active that year. The other didactic activities (theses, stages, etc.) are proposed either by the teachers or by the students themselves. (For simplicity, from now on we will consider all didactic activities as part of the syllabus.)

Student careers We see a student's career $\mathcal{C}$ as an ordered list of didactic activities performed, each tagged with the matriculation year of the student when he/she performed it, —which has to be greater or equal than the minimum matriculation year of the didactic activity. We say that $\mathcal{C}$ is *on* a set of didactic activities $\mathcal{A}$ if all the didactic activities in $\mathcal{C}$ are in $\mathcal{A}$. When matriculating, a student is given an empty career and is registered to the first matriculation year. Each time a student performs a didactic activity (passes an exam, presents successfully a seminar, defends a thesis, etc.), this activity is appended to the student career, together with the student's current matriculation year. If a didactic activity is a prerequisite of another, the former must occur before the latter. For instance,

$$\mathcal{C} = \{\langle GEO, 1\rangle, \langle CSI, 2\rangle, \langle CSII, 2\rangle, \langle OOP, 2\rangle\} \tag{1}$$

is a career on the set of courses in Figure 1.

Ordinances and Rules When matriculating, a student is associated to an *ordinance* ("ordinamento didattico"), which states the regulations for the student's career, that is, the goals to achieve and the constraints to satisfy for graduation. In the project model this is represented by a set of formal rules for the student career. Some examples of rules are given in Figure 2. From a syntactic viewpoint, each rule is built on the following components:

- a *quantity*, in the form *at least*|*at most* N. N is an integer, called the *bound*;
- a *unit of measure*, in the form *credits*|*units*;
- an *activity type*. The values correspond to those of the didactic activities, plus the value *any*, which matches all values;
- a *scope*, given by a *scope type*, in the form *subject areas*|*courses*|*all*, and a *scope list*, which is a list of elements of scope type. If the latter is *all* ("any scope"), then no scope list is provided.

	Quantity	Unit of Measure	Activity Type	Scope Type	Scope List
r_1 :	*at least* 180	*credits*	*any*	*all*	
r_2 :	*at least* 90	*credits*	*course*	*subject areas*	{MAT01,...,MAT08}
r_3 :	*at least* 38	*credits*	*course*	*subject areas*	{FIS01,...,FIS08, INF01,ING-INF05}
r_4 :	*at least* 10	*credits*	*course*	*subject areas*	{INF01,ING-INF05}
r_5 :	*at least* 5	*credits*	*thesis*	*all*	
r_6 :	*at least* 9	*credits*	*stage*	*all*	
r_7 :	*at least* 2	*units*	*course*	*courses*	$\{CSI, CSII\}$
r_8 :	*at most* 15	*credits*	*course*	*subject areas*	{INF01,ING-INF05}
r_9 :	...	...	...	...	...

Fig. 2. Examples of Rules.

We call a rule *explicit* if its scope type is *courses*, *implicit* otherwise. Explicit rules —e.g., r_7 in Figure 2— make explicit reference to course codes, which have to be defined a priori wrt. the rules themselves. Implicit rules —e.g., r_2, r_3, r_4 and r_8 in Figure 2— make no explicit reference to courses.

From a semantic viewpoint, each rule tells a student the amount of didactic activities of given kind and scope he/she has to cash into his/her career to achieve graduation. For instance, the intuitive meaning of the rule set in Figure 2 is that, a student has to cash into his/her career, respectively:

R_1: at least 180 credits on the whole (in any activities of any scope), of which:
R_2: at least 90 credits in courses in the subject area(s) $\{MAT01, ..., MAT08\}$,
R_3: at least 38 credits in courses in subject area(s) {FIS01, ..., ING-INF05}, ...

Notice the usage of the expression "of which". This applies when the scope and activity type of a rule r_i are subsets of the scope and activity type of another rule r_j, and the bound of r_i is smaller than the bound of r_j. (Remarkably, explicit scopes are treated as subsets of the scopes given by the corresponding subject areas; e.g., "*courses* $\{CSI, CSII\}$" is treated as a subset of "*subject areas* $\{INF01\}$".) Some rules with explicit scopes —like, e.g., r_7— can be used to force the student to perform some courses. We call such courses *mandatory*.

We divide the rules into *goal rules* and *constraint rules*. We call a goal rule any rule in the form "at least N ...". (Typically most rules in ordinances — often all— are goals.) A career *satisfies* a goal rule when the sum of units of the activities of the desired type and scope performed is greater or equal than the bound of the rule. For instance, the career (1) satisfies rule r_4 in Figure 2, as the 2nd, 3rd and 4th courses provide $5 + 5 + 5 \geq 10$ credits in the subject area *INF01*. The goal rules are not satisfied at the beginning of a career and must be satisfied when the student graduates. When a career satisfies a goal, all extensions of that career satisfy it.

We call a constraint rule any rule in the form "at most N ...". (Typically very few rules in ordinances —often none— are constraints.) A career *violates* a constraint rule when the sum of units of the activities of the desired type and scope performed is greater than the bound of the rule. For instance, in the career (1), the student cannot pass the exam of the sixth course of Figure 1, because doing so it would violate rule r_8 in Figure 2, as the courses 2, 3, 4 and 5 would provide $5 + 5 + 5 + 5 > 15$ credits in the subject area *INF01*. The constraint rules are not violated at the beginning of a career and none should be violated until the student graduates. If a career violated a constraint, all extensions of that career would violate it.

Given a career $\mathcal{C}$ and a rule set $\mathcal{R}$, we say that:

(i) $\mathcal{C}$ *violates* $\mathcal{R}$ if it violates one constraint rule $c_j \in \mathcal{R}$;
(ii) $\mathcal{C}$ *satisfies* $\mathcal{R}$ if it satisfies every goal rule $g_k \in \mathcal{R}$ and violates no constraint rule $c_j \in \mathcal{R}$.

$\mathcal{C}$ neither satisfies nor violates $\mathcal{R}$ if none of the two conditions above holds. Notice that a rule set is interpreted as a conjunction of rules.

2.2 A model for the problems

Career planning/synthesis A student has to plan his/her career when matriculating, and possibly to re-plan it every new year's registration, according to the rules he/she is given, the activities available, plus his/her own *desiderata*. (In many universities, such a plan must be presented and renewed explicitly.) Desiderata are mainly represented in terms of career elements $\langle a_i, y_i \rangle$ —e.g., $\langle OOP, 2 \rangle$ means "I want to pass OOP at the second matriculation year"— which we call *self-imposed didactic activities*. It is also possible to add to the rules of the ordinance self-imposed rules like, e.g., "At most 180 credits any all".

Because of the overlapping rule scopes, the different didactic activity types and credit weights, and the high number of degrees of freedom, the planning task may be rather complicate, and can cause mistakes. Thus, it is highly desirable to provide for the students a sort of *electronic advisor of studies* (EAS), an interactive support tool interfaced with the information system which, at each step,

(a) displays the student's current career and status wrt. the rule set and self-imposed didactic activities, and the complete list of activities available;
(b) allows the student to input and edit his/her own set of desiderata, and simulate interactively the evolution of his/her career
(c) can *synthesize automatically* careers which satisfy both the rule set and the desiderata.

The EAS can be used also by a secretary to verify the correctness of a complete student's career wrt. the ordinance rules.

While the steps (a) and (b) can be implemented with the standard technology of information systems, the step (c) can not, and it requires an external tool. The kernel of such a tool is an exhaustive search engine which, given the syllabus $\mathcal{A}$, the rule set $\mathcal{R}$, the current career $\mathcal{C}$, plus the student's set of self-imposed didactic activities $\mathcal{A}'$, searches for a career $\mathcal{C}'$, extending $\mathcal{C}$ and including all elements of $\mathcal{A}'$, which satisfies $\mathcal{R}$. This can be a very hard task, as it may require exploring all possible career combinations before finding one, or that there is none.

Remark 1. A syllabus lists only the courses which are taught the current year, and makes no statement about the future. Thus, any plan for a student's future career is necessarily based on the implicit assumption that the syllabus will not be modified in the future years. Luckily, the course list of the syllabus typically does not change much from year to year and, if it does, changes concern only with minor optional courses. (Mandatory courses are set by ordinance rules.) In general, in this paper we assume that syllabi are static. We are currently experimenting an implementation with dynamic syllabi. □

Verifying the coherence of a syllabus wrt. a rule set Defining the rule set of an ordinance, and defining new syllabi, are hard tasks. In fact, the rule set of an ordinance constrains not only the student careers, but also the syllabi. For instance, if the rule set defines some mandatory courses, the syllabi are forced to provide such courses; e.g., the goal r_7 of Figure 2 forces all syllabi to provide a couple of courses with code CSI and CSII, and corresponding subject area INF01. More generally, syllabi must always be *coherent* with the rule set, that is, they must always provide enough courses to give to every students who have never violated a constraint a chance to complete his/her degree course and graduate. We formulate the latter fact as follows.

Definition 1. Given a rule set $\mathcal{R}$ and set of didactic activities $\mathcal{A}$, we say that $\mathcal{A}$ is *coherent* wrt. $\mathcal{R}$ if it is always the case that, for every student's partial career $\mathcal{C}$ on $\mathcal{A}$ which does not violate $\mathcal{R}$, there exists at least one career $\mathcal{C}'$ on $\mathcal{A}$ extending $\mathcal{C}$ which satisfies $\mathcal{R}$.

Verifying exhaustively the coherence of the syllabi is in many cases out of the reach of a human mind. Thus, it would be highly desirable to provide a support tool able to verify it automatically.

In general, verifying the coherence of syllabi with the rule sets is by far out of the reach of standard technology of information systems, for it requires a (double) exhaustive search engine. In fact, the basic step is similar to that of the EAS, that is, to search for a new career $\mathcal{C}'$ extending $\mathcal{C}$ which satisfies $\mathcal{R}$. As before, this means exploring up to all possible career combinations before finding one. Much worse, this must be done *for every student's partial career* $\mathcal{C}$, which requires exploring up to all possible combinations.

Remark 2. One may wonder whether Definition 1 is too restrictive. Consider, for instance, the case in which the courses *OOP* and *AI* in Figure 1 had no prerequisite. Then the partial career $\mathcal{C} = \{OOP, AI\}$ would not violate r_8, but there would be no career $\mathcal{C}'$ extending $\mathcal{C}$ satisfying r_7 without violating r_8. Thus, according to Definition 1, the syllabus would not be considered coherent with the rule set. On the other hand, one can reply that, given a constraint c_j, if we provide a set of didactic activities $\{a_1, ..., a_k\}$ containing mandatory courses which is big enough to violate c_j, we should also introduce prerequisites to force the student to perform all the mandatory courses before violating the constraint. We will further discuss the topic in Section 4. $\square$

3 Formalization into Model Checking

The main idea of this paper is to encode and solve the problems of the synthesis of student careers and of the verification the coherence of the syllabi wrt. rule sets, as CTL model checking problems. For lack of space, we omit any description of CTL and OBDD-based CTL model checking, which can be found in, e.g., [Clarke *et al.*1986; McMillan1993].

3.1 Career evolutions as a finite state machine

We represent all the possible evolutions of a student's career within a given set of didactic activities as a finite state machine (FSM from now on). Consider Figure 3. Broadly speaking, a state is characterized by the set of the activities performed so far. In the initial state no activity has been performed; each transition represents the performance of a new activity, and adds it to the career.

More in detail, let $\mathcal{A}$ be the set of didactic activities a_i's of a given syllabus; let $\mathcal{R}$ be the rule set of a given ordinance; let g_j and c_k denote the goal rules and the constraint rules in $\mathcal{R}$ respectively. We represent all the possible evolutions of a student's career within $\mathcal{A}$ as a FSM $\mathcal{M}$.

The state variables of $\mathcal{M}$ are given by:

- an array $\mathbf{v}$ of booleans, one for each didactic activity a_i in $\mathcal{A}$, such that $\mathbf{v}[i]$ is `true` if and only if the didactic activity a_i has been performed.
- a bounded integer $y \in \{1, ..., y_{max}\}$, representing the student's current matriculation year;
- an array $\mathbf{b}$ of bounded integers, one for each rule r_j in $\mathcal{R}$, s.t. $\mathbf{b}[j]$ is the sum of the credit/unit weights of the activities in the current career which match the type and scope of r_j.
- an array of booleans $\mathbf{p}$, one for each a_i in $\mathcal{A}$, s.t $\mathbf{p}[i]$ is `true` if and only if a_i satisfies its matriculation year and prerequisite constraints.

Notice that $\mathbf{b}$ and $\mathbf{p}$ are not state variables in the strict sense, as their values derive deterministically from the values of $\mathbf{v}$ and y. A state is univocally denoted by the values of $\mathbf{v}$ and y, so that the size of the state space is upper-bounded by $y_{max} \cdot 2^{||\mathcal{A}||}$, $||\mathcal{A}||$ being the number of didactic activities in $\mathcal{A}$.

In the initial state of the FSM $\mathcal{M}$, the matriculation year y is 1; every boolean $\mathbf{v}[i]$s is set to `false` (no activity performed); every bounded integer $\mathbf{b}[j]$ is set to 0; for each a_i, $\mathbf{p}[i]$ is set to `true` if a_i's minimum matriculation year is 1 and a_i has no prerequisite courses, to `false` otherwise.

The transition relation of $\mathcal{M}$ is defined in such a way that

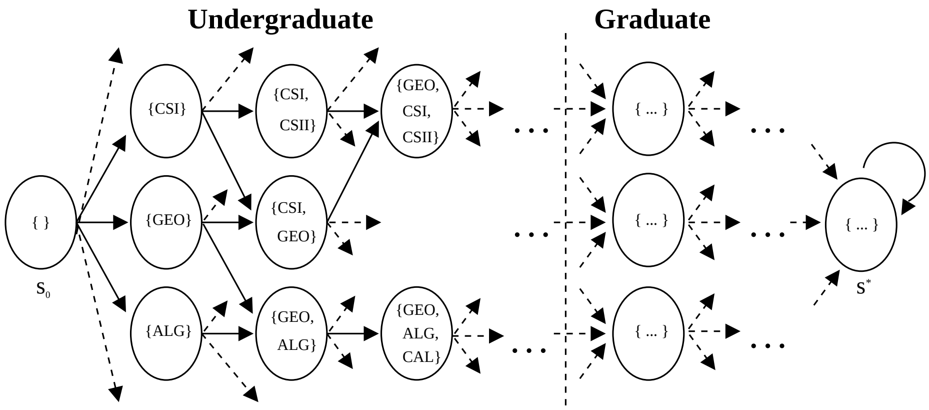

Fig. 3. The FSM representing all the possible evolutions of a student's career within a given syllabus. Within each state, "{}" is the set of didactic activities performed. (For simplicity, information regarding matriculation years and prerequisites is ignored.)

- one boolean $\mathbf{v}[i]$, corresponding to the performed activity a_i, passes from `false` to `true`, the others keep their values. If all $\mathbf{v}[i]$'s are `true`, then they all keep their values;
- y may either keep its value or be incremented by 1. If y equals its bound y_{max}, then it is not changed. If all $\mathbf{v}[i]$'s are `true` and y is smaller than the bound, it is incremented by 1;

The other variables are automatically updated from the new values of $\mathbf{v}$ and y in the following way:

- for each rule r_j in $\mathcal{R}$, if the performed activity a_i matches the type and scope of r_j, then $\mathbf{b}[j]$ is incremented by the weight of a_i, otherwise $\mathbf{b}[j]$ keeps its value.
- for each a_k, $\mathbf{p}[k]$ is set to `true` if a_k's minimum matriculation year is smaller or equal y, and a_k has no prerequisite courses a_l s.t. $\mathbf{v}[l]$ is `false`, it is set to `false` otherwise.

The goal and constraint rules in $\mathcal{R}$ are the atomic propositions of the CTL formulas representing the specifications. Thus, for every state s of $\mathcal{M}$ and for every goal rule g_k and constraint rule c_j in $\mathcal{R}$, we say that

$$\begin{aligned} \mathcal{M}, s \models g_k &\iff \mathbf{b}[k](s) \geq \mathit{bound}(g_k), \\ \mathcal{M}, s \models c_j &\iff \mathbf{b}[j](s) \leq \mathit{bound}(c_j), \end{aligned} \tag{2}$$

where $\mathbf{b}[i](s)$ denotes the value of $\mathbf{b}[i]$ in s. Intuitively, this means that the goal rule g_k "at least N credits ...." is true in the state s of the FSM $\mathcal{M}$ if and only if, in the state s, the amount of credits $\mathbf{b}[k]$ matching the scope and type of g_k is greater or equal than N, and that the constraint rule c_j "at most N credits ...." is true in the state s of the FSM $\mathcal{M}$ if and only if, in the state s, the amount of credits $\mathbf{b}[j]$ matching the scope and type of c_j is smaller or equal than N.

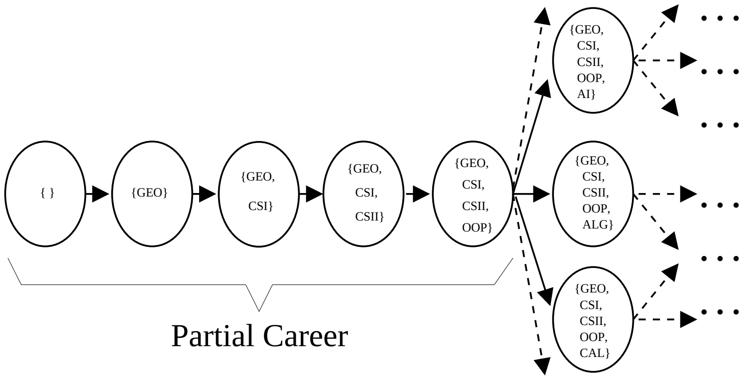

Fig. 4. The FSM representing all the possible evolutions of a student's career starting from a given partial career. (For simplicity, information regarding matriculation years and prerequisites is ignored.)

To handle the case when the evolutions starts from a given partial career $\mathcal{C}'$, we modify $\mathcal{M}$ by forcing an initial deterministic behavior until the given career $\mathcal{C}$ is emulated. This is represented in Figure 4. Let $\mathcal{C}$ be the career $\{\langle a_{i_1}, y_{i_1}\rangle, ..., \langle a_{i_N}, y_{i_N}\rangle\}$. At the k-th step, $k \leq N$, $\mathcal{M}$ "chooses" deterministically the didactic activity a_{i_k}, sets $\mathbf{v}[i_k]$ to `true` and y to y_{i_k}, and updates $\mathbf{b}$ and $\mathbf{p}$ accordingly. After the Nth step, $\mathcal{M}$ starts behaving in the usual non-deterministic way. To handle the presence of self-imposed didactic activities $\langle a_i, y_i\rangle$'s, we modify $\mathcal{M}$ by forcing the deterministic choice of a_i as soon as possible when $y \geq y_i$. [3]

Property 1 (Monotonicity). Assume `true` > `false`. Let s and s' denote two generic states of the FSM $\mathcal{M}$ described above such that s' is a successor of s. Let x and x' denote the values of the generic variable x in s and s' respectively. Then we always have that, for every i and j, $y' \geq y$, $\mathbf{v}[i]' \geq \mathbf{v}[i]$, $\mathbf{b}[j]' \geq \mathbf{b}[j]$, $\mathbf{p}[i]' \geq \mathbf{p}[i]$. We say that $\mathcal{M}$ is *monotonic.* □

A consequence of the definition of $\mathcal{M}$ is that, except for the very last state s^* where all $\mathbf{v}[i]$s are `true` and y equals its bound, it is always the case that $y' > y$ or $\mathbf{v}[i]' > \mathbf{v}[i]$ for one i, that is, the monotonicity of $\mathcal{M}$ is "strict". Thus we have no loops except than in s^*, which is the successor of itself (see Figure 3).

From (2) and Property 1 we have that, for every goal rule g_k and constraint rule c_j in $\mathcal{R}$,

$$\begin{array}{lll} \mathcal{M}, s \models g_k & \Longrightarrow & \mathcal{M}, s' \models g_k, \\ \mathcal{M}, s \not\models c_j & \Longrightarrow & \mathcal{M}, s' \not\models c_j. \end{array} \tag{3}$$

This matches the intuitive statements of Section 2.1: when a career satisfies a goal, all extensions of that career satisfy it, and when a career violates a constraint, all extensions of that career violate it.

[3] This means interpreting $\langle a_i, y_i\rangle$ as "I want to perform a_i *as soon as possible* in the y_i-th matriculation year". Anyway, as self-imposed didactic activities can be used only in career synthesis, this is not a significant restriction.

3.2 Encodings of the different problems

Let $C =_{def} \bigwedge_j c_j$ and $G =_{def} \bigwedge_k g_k$, where c_j and g_k denote the constraint and goals in $\mathcal{R}$. Let s_0 denote the initial state in $\mathcal{M}$. We represent the different kinds of problems described in Section 2.2 as CTL model checking problems by means of different specification formulas.

Career's synthesis The problem of synthesizing a career matching the rules and desiderata is a typical *reachability* problem: given a set of initial states Δ, a set of goal states Γ (the states which verify all goals and constraints) and a transition relation T, find a path $s_0, ..., s_n$ such that $s_o \in \Delta$, $s_n \in \Gamma$ and $T(s_{i-1}, s_i)$ holds for every $i \in \{1, ..., n\}$. Following the approach of [Cimatti *et al.*1998], we model the problem as:

$$\mathcal{M}, s_0 \models \mathbf{AG}\neg(C \wedge G). \tag{4}$$

The CTL specification formula means "invariantly, at least one rule is false". The model checker tries to verify exhaustively this property, and, when it finds a counter-example (a state s_n where all rules are true) it returns a path π leading to it. The property (3) guarantees that, if a constraint is not violated in s_n, then it is not violated in all the states of the path. Thus a path π returned represents the progression of a career which satisfies all goals and violates no constraint. If a breadth-first search strategy is used (as in standard OBDD-based CTL model checking), then the length of the career returned is minimal. If the synthesis starts from a partial career $\mathcal{C}'$, or it contains self-imposed didactic activities, $\mathcal{M}$ is modified as described in the previous section.

Verifying the coherence of a syllabus Following Definition 1, we model the problem of verifying the coherence of a syllabus as follows:

$$\mathcal{M}, s_0 \models \mathbf{AG}(C \rightarrow \mathbf{E}(C\mathbf{U}(C \wedge G))). \tag{5}$$

The CTL specification formula means "invariantly, if in a state s all constraints are true, then there exist a path π starting from s in which all constraints are always true and eventually all goals become true." The intermediate state s represents that of the partial career $\mathcal{C}$ in Definition 1, and the path π represents the career prosecution $\mathcal{C}'/\mathcal{C}$. Because of the monotonicity property (3), (5) can be simplified into:

$$\mathcal{M}, s_0 \models \mathbf{AG}(C \rightarrow \mathbf{EF}(C \wedge G)). \tag{6}$$

Moreover, as the goals are monotonic (3) and the FSM has no loop except for the one in the final state s^*, if there is no constraint rule, then (6) can be further be simplified into:

$$\mathcal{M}, s_0 \models \mathbf{AF}\ G, \tag{7}$$

which means "for all paths, eventually all goals become true".

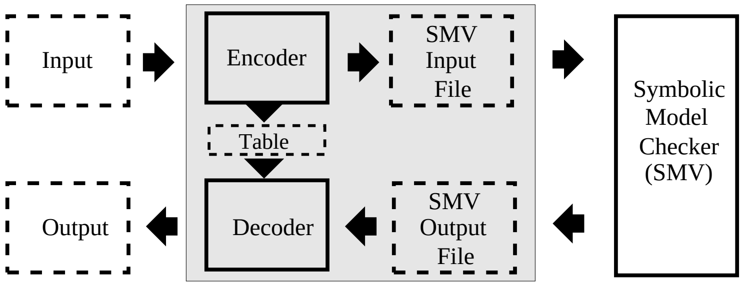

Fig. 5. Schema of the prototype tool.

4 A prototype implementation

The project information system will be ready no earlier than the second half of 2001. In the meantime, to verify the feasibility of the approach, we have implemented a simple prototype tool interfaced with a new version of the SMV symbolic model checker [McMillan1993]. The tool provides a set of commands and options to perform the tasks described in Section 3. It takes as input a set of didactic activities $\mathcal{A}$, a rule set $\mathcal{R}$, and, in case of career synthesis, an optional partial career $\mathcal{C}$ and an optional set of self-imposed didactic activities $\mathcal{A}'$. The output depends of the kind of problem addressed: in case of career synthesis, either it returns a career $\mathcal{C}'$ extending $\mathcal{C}$ and including all elements of $\mathcal{A}'$ which satisfies $\mathcal{R}$, or it fails; in case of syllabus coherence verification, it returns `true` if $\mathcal{A}$ is coherent wrt. $\mathcal{R}$, `false` otherwise.

The schema of the tool is reported in Figure 5. An ENCODER generates from the input a SMV input file describing the FSM $\mathcal{M}$ and the CTL specification formula, as in Section 3, plus a TABLE, keeping track of the symbol encodings. A DECODER converts the SMV output file in a readable output format using the TABLE; in particular, it converts an output path of states into a career. The search engine of the tool is the symbolic model checker, which is used as a blackbox. The choice for an OBDD-based CTL model checker was forced by the particular formalization of the problems, as in Section 3, and by the very regular structure and high symmetry of the FSM $\mathcal{M}$.

The tool provides some options to improve its performances. First, we notice that a significant source of resources consumption is the presence of lots of counters $\mathbf{b}[j]$: OBDD-based model checkers handle with difficulty (bounded) integers, as they have to be encoded into their bitwise representations. Thus, if an apposite option is set, the tool applies the following reduction: if the greatest common divisor (GCD) n of the credit weights of all the didactic activities in $\mathcal{A}$ is strictly greater than 1, then all the credit values are divided by n by the ENCODER and the results are re-multiplied by n by the DECODER. This allows the model checker to handle smaller integer values. For the goal and constraint rule bounds b's, it is considered respectively the ceiling $\lceil b/n \rceil$ and the floor $\lfloor b/n \rfloor$. In fact, e.g., if the weights of all didactic activities are multiple of 5, "at least 9" is equivalent to "at least 10", and "at most 7" is equivalent to "at most 5". We call this option, *GCD*.

Moreover, the number of possible paths may be huge, and most are groupwise identical modulo order permutations. We notice that non-mandatory didactic activities cannot be prerequisites for mandatory courses (this fact is checked by the ENCODER). Thus, one may want to restrict the search to careers in which mandatory courses are performed as soon as possible —that is, at the beginning of their corresponding minimum matriculation year. Moreover, from the students' viewpoint, non-mandatory didactic activities are the only matter of choice in a career, so that, once the mandatory courses are clustered at the beginning of their matriculation years, their relative order within each cluster can be considered insignificant. Thus, one may want to impose a fixed order among the clustered mandatory courses, which is compatible with the prerequisites. We call this option, *Restrict*. Notice that the Restrict option allows the model checker to avoid situations like the one highlighted in Remark 2.

Finally, if an apposite option is set, all rules stating mandatory courses — like, e.g., r_7 in Figure 2— can be merged into one rule, requiring thus only one counter. We call this option, *Merge*.

5 Preliminary empirical results

The reform will come into force in the year 2001-2002, so that so far the universities have not yet presented their post-reform ordinances and syllabi. Thus, to verify the feasibility of the approach, we have modeled as a test-case the proposal of post-reform ordinance and syllabus of the Mathematics and Physics degree of the University of Trento. [4] Both degree course ordinances require 180 credits on the whole, of which at least 5 credits are for a thesis and 9 are for a stage, with no scope restriction. Both rule sets contain no constraint rule. The Math syllabus offers 55 5-credit courses, among which 22 are mandatory; the Physics syllabus offers 39 5-credit courses, among which 25 are mandatory.

As stages and theses are not explicitly inserted in the syllabi, we have added them in two distinct ways: in the first ($Math_1$ and $Phys_1$) the syllabi are added one 5-credit didactic activity of type `thesis` and a 10-credit one of type `stage`; in the second ($Math_2$ and $Phys_2$) the syllabi are added one 5-credit and one 6-credit didactic activity of type `thesis` and one 9-credit and one 10-credit didactic activity of type `stage`; a constraint rule "at most 1 unit [of type] thesis" is added in order to cope with the uniqueness of the thesis.

The results of the empirical tests are summarized in Table 1. (All tests have been obtained by running the model checker on a bi-processor PC `PentiumIII 667MHz 1GB RAM` with `Debian Linux`; the RAM consumption and CPU time required by the ENCODER and the DECODER are negligible wrt. those of the model checker.) We have considered both the problems of career synthesis and syllabus coherence verification. For all problems, we have analyzed all possible

[4] Available at `http://www-math.science.unitn.it/CCLM/` (in Italian). As they are just proposals, they may eventually change wrt. the current version (October 2000).

GCD	off	off	off	off	on	on	on	on	ordinance/ syllabus	Problem
Restrict	off	off	on	on	off	off	on	on		
Merge	off	on	off	on	off	on	off	on		
BDD Nodes	???	25M	288K	244K	???	27M	219K	198K	$Math_1$	Career synthesis
CPU Time		26'	11"	6"		22'	6"	4,5"	[GCD=5]	
BDD Nodes	???	???	314K	254K	???	???	314K	254K	$Math_2$	
CPU Time			11"	7"			11"	7"	[GCD=1]	
BDD Nodes	???	825K	97K	65K	???	801K	107K	61K	$Phys_1$	
CPU Time		27"	5"	2,5"		26"	2,5"	1,5"	[GCD=5]	
BDD Nodes	???	1,5M	122K	103K	???	1,5M	122K	103K	$Phys_2$	
CPU Time		51"	5"	3"		51"	5"	3"	[GCD=1]	
BDD Nodes	???	???	10M	4,3M	???	???	2,9M	1,0M	$Math_1$	Syllabus verification
CPU Time			488"	300"			75"	40"	[GCD=5]	
BDD Nodes	???	???	7,2M	9,2M	???	???	7,2M	9,2M	$Math_2$	
CPU Time			24'	24'			24'	24'	[GCD=1]	
BDD Nodes	???	3,4M	258K	111K	???	3,7M	253K	96K	$Phys_1$	
CPU Time		149"	19"	15"		127"	3,5"	2"	[GCD=5]	
BDD Nodes	???	15M	499K	353K	???	15M	499K	353K	$Phys_2$	
CPU Time		28'	36"	31"		28'	36"	31"	[GCD=1]	

Table 1. The results of the tests. "???" means "Exceeding 600MB RAM consumption". (Here "K" and "M" denote 10^3 and 10^6 rather than 2^{10} and 2^{20}.)

combinations of the *GCD*, *Restrict* and *Merge* options. (For $Math_2$ and $Phys_2$, the GCD option is ineffective, so that the values in the left half of the table are pairwise identical to those of the right half.)

On one hand, we notice that the problems are intrinsically very hard: with all the options disabled, no problem can be solved within 600MB RAM consumption, as the size of the state space, and thus of the BDD's, tend to explode. The same happens when both Restrict and Merge options are off. On the other hand, with all the options set, all problems are well at the reach of the model checker, mostly solved in a bunch of seconds. In between, we notice that:

- with the GCD option on, when applicable ($Math_1$ and $Phys_1$), there are mostly sensible, but not dramatic, performance improvements. In fact, it reduces the number of boolean variables necessary to encode bounded integers, but the reduction is only logarithmic with the value of the GCD;
- with the Restrict option on, the improvements are very relevant. In fact, it reduces the number of non-deterministic choices and restricts the search only to non-mandatory didactic activities, causing a very relevant reduction of the size of the state space.
- with the Merge option on, there are mostly relevant performance improvements. In fact, it allows for a reduction of the number of the counters —i.e., of the boolean variables encoding it— and reduces the size of the corresponding BDD.

We notice that $Math_2$ and $Phys_2$ are always harder or much harder than $Math_1$ and $Phys_1$ respectively. The motivations for this fact are manifold: first, we

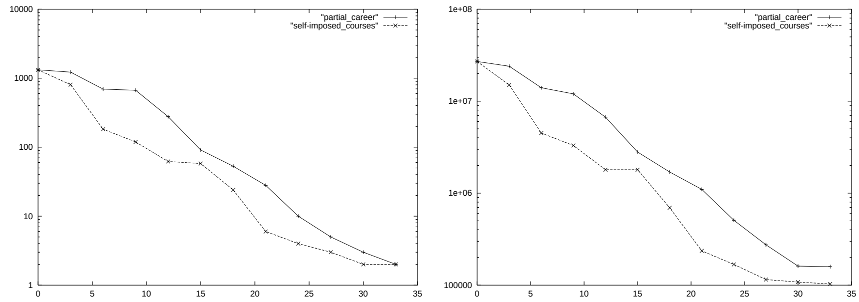

Fig. 6. Results obtained with $Math_1$, GCD=on, Merge=on and Restrict=off, adding either a partial career or a set of self-imposed didactic activities of increasing size. Y axis: (Left) CPU time in seconds; (Right) number of BDD nodes. X axis: number of didactic activities inserted.

have two more non-mandatory didactic activities; second, the new 6-credits and 9-credits didactic activities hinder the applicability of the GCD option ($gcd(5, 6, 9) = 1$); finally, for the syllabus verification problem, with $Math_1$ and $Phys_1$ we use the simplified encoding (7), while with $Math_2$ and $Phys_2$ the constraint rule added forces the usage of the much harder encoding (6).

As a side observation, we also notice that the problem of career synthesis is always easier or much easier than the corresponding problem of syllabus verification. In fact, when both problems have solution, the latter requires an exhaustive exploration of the whole search space, while the former can stop when it finds one path.

For the problem of career synthesis, we wonder how the efficiency changes when we add partial careers and self-imposed didactic activities. To provide an intuition, in Figure 6 we have taken the problem $Math_1$ with GCD=on, Merge=on and Restrict=off, and we have added either a partial career or a set of self-imposed didactic activities of increasing size. Both the CPU times and number of OBDD nodes decrease significantly with the size of both the partial career and the set of self-imposed didactic activities. In fact, as with Restrict=on, the pre-compiled presence of the input didactic activities in the career reduces the number of non-deterministic choices and restricts thus the search, causing a relevant reduction of the size of the state space.

6 Ongoing and future work

The currently implemented prototype is actually more complex than we have described. First, it can handle multiple choices for the credit weight of one didactic activity. E.g., for theses, this prevents introducing extra constraints like the one in Section 5, which is the main source of the performance gaps between $Math_1$, $Phis_1$ and $Math_2$, $Phis_2$. Second, it can also handle dynamic syllabi. This feature, which is still at experimental level and has to be further investigated, requires handling some extra information, like the range of years in which an exam can be passed and the equivalence relations among courses. The key future research step, however, will be to pass to a direct integration with a symbolic model checker instead of the current blackbox usage. This should allow for customizing the model checking engine to exploit the very peculiar features of the FSM, and for improving the level of interactivity of the career synthesis process.

We may wonder whether our approach is the right one for this application. For the problem of syllabus coherence verification, we need a language which, like CTL, is expressive enough to represent boolean and temporal information as well as constraints on bounded integers, and we need a search technique which, like OBDD-based Model Checking, is able to exploit the very regular structure of the FSM to cope with the combinatorial explosion of the state space. Therefore, although we cannot exclude a priori the existence and effectiveness of other approaches, we believe that symbolic model checking is a natural and very effective way to encode this problem.

For career synthesis, the problem is significantly simpler, and we are considering also other approaches like planning combined with linear programming [Wolfman & Weld1999] and SAT-based model checking [Biere *et al.*1999], with some tricks from [Giunchiglia *et al.*1998] to exploit the dependencies among the state variables of the FSM.

References

[Biere *et al.*1999] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. In *Proc. TACAS'99*, pages 193–207, 1999.

[Cimatti *et al.*1998] A. Cimatti, M. Roveri, and P. Traverso. Strong Planning in Non-Deterministic Domains via Model Checking. In *Proc. Fourth International Conference on Artificial Intelligence Planning Systems (AIPS-98)*, 1998.

[Clarke *et al.*1986] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[Giunchiglia *et al.*1998] E. Giunchiglia, A. Massarotto, and R. Sebastiani. Act, and the Rest Will Follow: Exploiting Determinism in Planning as Satisfiability. In *Proc. AAAI'98*, pages 948–953, 1998.

[McMillan1993] K. McMillan. *Symbolic Model Checking*. Kluwer Acad. Publ., 1993.

[MURST2000a] MURST. Regolamento in materia di autonomia didattica degli atenei (Regulations on the didactic autonomy of universities). *Gazzetta Ufficiale*, 2, January 2000. In Italian. Available at http://www.murst.it/regolame/1999/adq.htm.

[MURST2000b] MURST. Rideterminazione dei settori scientifico-disciplinari (Redefinition of subject areas). *Gazzetta Ufficiale*, 3, January 2000. In Italian. Available at http://www.murst.it/atti/2000/dSettori.htm.

[Wolfman & Weld1999] S. Wolfman and D. Weld. The LPSAT Engine & its Application to Resource Planning. In *Proc. IJCAI*, 1999.