Designing Intelligent Tutoring Systems: A multiagent Planning Approach

Roger Nkambou and Froduald Kabanza
Département de Mathématiques et Informatique
Université de Sherbrooke, Sherbrooke, J1K 2R1
{roger.nkambou, froduald.kabanza}@dmi.usherb.ca

Abstract

Intelligent Tutoring Systems (ITS) are computer aided intelligent learning tools. Most recent architectures of these systems have focussed on the tutor or curriculum components, but with little attention being paid to planning and intelligent collaboration between the different components. In this paper, we propose several improvements by describing a new architecture that involves sophisticated planning processes at different levels of the ITS processing and by decomposing the tutor into two different components, one specialized in the tutorial actions planning and the other tailored for the generation of multimedia presentations. Moreover, the user interface is made more robust and flexible thanks in part to the use of planning techniques. These improvements are built on Nkambou, Gauthier and Lefebvre's architecture, but we believe that the most of the important ideas discussed herein may also be exploited in other architectures

1. Introduction

Over the last ten years, intensive research has been conducted on Intelligent Tutoring Systems (ITS), several of them focusing on architectural issues. The first proposed ITS architectures and included four main components (Burn and Caps, 1988): a curriculum module, a student model, a tutor (pedagogic model), and an interface between the student and the system. This basic architecture has since been extended by many researchers, including Bass, 1998; Choquet et al. 1998; Titter and Blessing 1998; Frasson et al. 1996, Nkambou and Gauthier, 1996; Both the architectures designed by Ritter and Blessing and Frasson et al. Are focused on the tutor. Bass architecture mainly improves the interface. Nkambou et al. mostly focus on the curriculum component.

We propose several improvements on the architecture of Nkambou, Gauthier and Lefebvre by involving sophisticated planning processes at different levels of the ITS processing and by decomposing the tutor into two different components, one specialized in the tutorial actions planning and the other tailored for the generation of multimedia presentations. Moreover, the user interface is made more robust and flexible thanks in part to the use of planning techniques. Although these improvements are built on Nkambou, Gauthier and Lefebvre's architecture, we believe that the most important ideas discussed herein may be exploited in other architecture. As such our contribution is not only improvement of a very specific architecture but also a discussion on the use of planning techniques in ITS in general.

Nkambou, Gauthier and Lefebvre's original architecture has three main components: curriculum, lesson planner, and tutor. It also includes the student model, didactic resource and the student interface.

Since our architecture extends the one from Nkambou, Gauthier and Lefebvre, it helps to remind first how their architecture is designed. Their ITS is composed of three main components: curriculum, lesson planner, and tutor, with three other components: the student model, didactic resource and the student interface.

The curriculum is comprised of two parts: a set of curriculum objects and a course object. Each curriculum object describes a subject matter from a domain, pedagogical and didactical point of view. A course object describes one particular course[1]. The attributes of a course (its summary and its preliminaries, among others) determine the intentions of the course builder (purposes of the teaching), and this, independently of a particular student using the ITS. The preliminaries of a course allow to set the targeted audience. The identification of such an audience helps the instructional designer to construct the curriculum. If the ITS detects that some preliminaries are not acquired (by a test to verify if the student knows the preliminaries or otherwise), then the student would have to acquire these preliminaries before starting the course. Of course, if the curriculum is sufficiently complete the ITS could by itself teach the missing preliminaries.

The lesson planner is responsible of the global aspects of individualized teaching. It proposes teaching tasks (to motivate the student, to recall previous knowledge, to formulate an objective, to present content, to assess the student...) for the tutor to realize.
The lesson rely on the course specification and takes into account the learner, as perceived through the portion of the student model that concerns the subject matter (objectives and pedagogical resources available). The lesson planner runs online so that it can re-planning courses whenever the tutor requires so.

The tutor realizes the teaching tasks by choosing and using pedagogical resources (teaching material), controls the teaching process and the dialogue with the student according to the selected tutoring mode (coaching, critiquing, …) replies to remediation needs and to help requests from the student, updates the student model.

The student model includes knowledge (system beliefs, student beliefs, inferred knowledge) on what the student know about the subject matter. It also contains student behaviour and preferences. The Student model is initialised when the student takes its first course and evolve along the learning process. Therefore, it is a global model that mimics the student during the training process. The Student model is central in an intelligent tutoring system and is used as the basis of several decision making in this system particularly for the individualisation of the learning process. It allows initialization, consultation and update of the information about the learner. The model also includes a propagation model able to control the evolution of knowledge about the learner (Nkambou, Lefebvre and Gauthier 1996).

Didactic resources are tactical means to support a learning session. It can be a demonstration, an exercise, a problem, a hypermedia or multimedia document, a HTML document. They contain their own management system (including evaluation of the resource execution).

[1] A choice of several courses can be offered to the learner, nevertheless we place ourselves in the situation where the learner has already chosen a course.

The interface assume the communication between ITS modules and the learner in a learning environment.

Figure 1 shows the overall interaction model of the system designed using OMT approach.

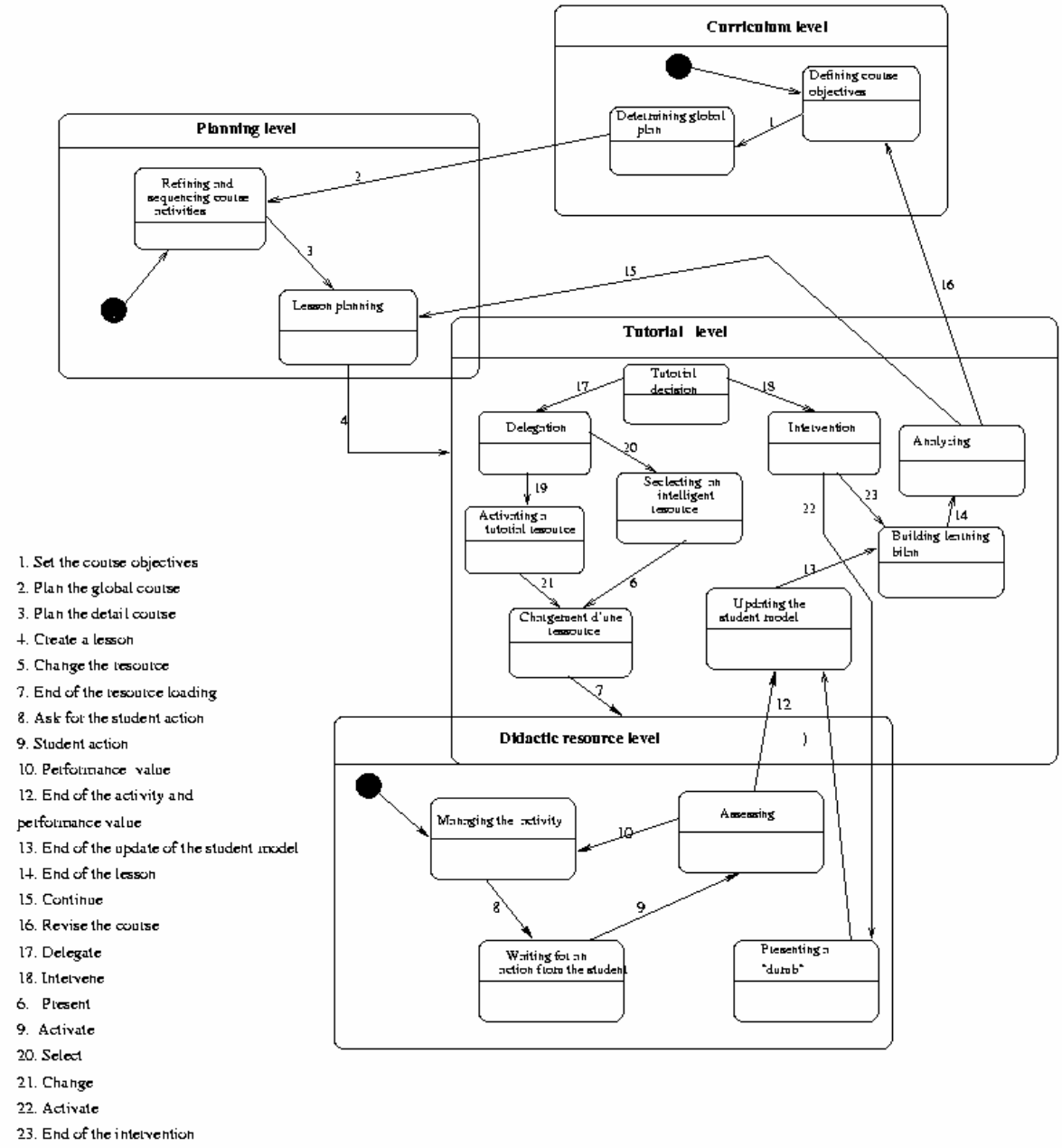

Figure 1: Statecharts modeling interaction between components

One problem with this architecture is the heterogeneity of the components. This makes it difficult to design the between components because one has to deal with different data representations in different components. This problem is addressed in our new architecture by observing that the curriculum, the lessor planner, the tutor are all concerned in a manner or another with the planning problem, that is, the problem of deciding in advance actions that will be executed in expected future situations in order to achieve a particular goal. Hence, we can see each component as basically a planning process, although each process operate at its own level of abstraction.

By "planning process", we mean any process that decide what action to do and when in order to achieve a particular goal. A set of decision rules for accomplishing a given goal is also called plan in the sense that decisions are computed by anticipating future interactions between the processes composing the system and its environment. In the case of the curriculum, the task is to plan individualized course. A basic action is to achieve a particular instructional objective; the goal is to mastery a set of concepts at a given levels. In the case of the lesson planner, the task is to generate the next concept or topic the system has to focus on and recommends general context of the achievement. A basic action is to include an introduction statement, or a reminder statement; the goal is to have a lesson plan that can permit to mastery a specific concept at a given level. In the case of a tutor, the task is to plan a set of basics tasks (activities) the students have to achieve in order to acquire knowledge determine by the planner and the way those tasks will be presented to the student. A basic action is "choosing a learning strategy"; the goal is to have the sequences of resources to activate in order to realize lessons goals. The tutor also generates the activities presentation and interaction with the student. Our new architecture is articulated around several planning agents, each in charge of a particular aspect of the ITS. As far as the design is concerned, the different planning agents may be implemented by the same planning algorithm, but with different input knowledge.

Another problem with the above ITS architecture is that planning tasks are currently done in an ad-hoc and scruffy way. The underlying planning algorithm is not bolstered by a clear model upon which one could rely to prove the correctness or optimality of generated plans. Furthermore, currently used planning approaches in ITS do not reason about temporal constraints associated with the learning process or timing constraints in the synchronization of different medias used to present a course. Rather, such constraints are dealt with heuristically outside the planning processes. Note that these limitations not only apply to the above architecture, but also to most existing ITS architectures.

A great deal of these planning limitations can be overcome by using well-known planning algorithms in the area of artificial intelligence (e.g., see Bacchus and Kabanza, 1996; Dean et al. 1995; Kabanza, Barbeau and St-Denis, 1997). These algorithms rely on well defined semantics that allow to prove the correctness and optimality of generated plans. On the other hand, AI planning research has been addressing various issues that are relevant to planning in ITS such as handling temporal constraints and reasoning about uncertain information. Some of this research has been applied to the automatic generation of multimedia presentations (André and Rist, 1996). As part of the tutor may consist in producing presentation, this last application is very relevant to tutor planning.

It is surprising that AI planning techniques have been so under-exploited in ITS, despite the acknowledged importance of planning in ITS. Our new architecture fills in this gap. It is not designed is a mere plug-in of AI planning algorithms into an existing ITS system. In fact, it also involves innovative ideas on the coordination between different planning processes. The research in AI planning provides us with quite good planning algorithms applicable to different planning tasks in ITS. But beyond the question of how to plan, there is the question of to coordinate several planning processes, running concurrently, at different levels of abstraction in an ITS. The problem of planning for concurrent or distributed agents has been addressed in the area of distributed artificial intelligence and multiagent systems, mostly for robotics systems (e.g., see Georgeff and Lansky, 1987; Spector and Hendler, 1992). While this provides a useful inspiration for concurrent planning in ITS, the solutions are not adequate because of the ITS domain involves very specific issues. The environment for an ITS involves for example a student, so the system must reason about a student model. This is different of the kind of reasoning a robot must make to manipulate parts, so we can expect different representation languages in both kind of applications and different reasoning strategies.

Our new architecture hinges on a multi-layer planning system in which several planning agents run concurrently, collaborating in order to prepare lessons and teach them to the learners. Since planning agents play an important role in our new architecture, it helps to pursue immediately with some general background on planning agents in artificial intelligence. Then, we will outline the new architecture by focusing only on its main components, leaving aside the details on how each component actually behave. Finally, we will describe one of the components with more details. Due to space limitations, the other components will not be detailed in this paper. Instead we will refer to an extended version.

2. Planning agents

Agents are one of the most important emerging concepts in computer science today. As the term suggests, an agent is a system that has some delegation capabilities, that is, a system that can accomplish various tasks on behalf of human users or of other systems. This is in accordance with the notion of an agent in everyday life such as bank tellers or mutual fund brokers. In fact, most computer programs have a delegation capability to some extent in the sense that they make calculations on behalf of users. Accordingly, some researchers do not hesitate to qualify such a simple program as the square root of a number as an agent, albeit its simplicity (Russel and Norvig, 1995). Other researchers and software developers believe that the term “agent” should be reserved to only those processes that have some sophisticated capabilities of reasoning about their environment and actions in order to communicate with other interacting processes or to achieve purposeful tasks (Sycara, 1998).

We subscribe to this last notion of an agent. We are interested in the design and implementation of agents that are processes capable of accomplishing autonomously various tasks on behalf of users or other processes, sometimes with various intelligent competencies, such as the ability to plan courses at different level of abstraction, or to learn from previous student and ITS interactions, and to plan multimedia presentation from planned courses and raw media materials.

Planning agents will play three main roles in our new ITS architecture. Firstly, the ITS will be able to accomplish tasks or deal with events for which it has not been explicitly programmed, by

planning on-line decision rules that coordinate more basic ITS processes so that they react correctly. Secondly, since planning agents yield decision rules that are constructively proven correct and optimal, the ITS reliability will become increased. Finally, planning agents will contribute to a flexible user interface for the ITS by enabling users to describe *what* they want the system to do (i.e., goals); then details on *how* the task should be done (i.e., plans) will be conveyed by decision rules that are computed by on-line planning processes.

Basic algorithms for planning agents can be defined using explicit exploration of tree of possible ITS and student actions (e.g., see (Bacchus and Kabanza, 1996)), the dynamic programming of Markov decision processes (e.g., see (Dean et al. 1995)) or control theoretical techniques (e.g., see (Barbeau, Kabanza and St-Denis, 1998)). All these approaches explore a search space to compute decision rules. The search space may, however, be differently structured, depending on formalisms used to model actions, goals, and environments, and search techniques. Since different formalisms and search techniques yield incomparable performances or expressiveness, this justifies research on hybrid approaches.

SIMPLAN (Simulation Based Planner) is a recently developed planning system that combines model checking and Markov decision theoretic techniques (Kabanza, 1999). It will be used as the basic planning system in our architecture. Although we opted for this particular planning system in order to fix a concrete development framework, the architecture of our ITS is open so that the planning algorithm underlying a particular planning agent may be replaced without jeopardizing the correct functioning of the system as a whole. Interfaces between various planning agents will be involved to abstract over differences that may exist between input or output formats for different planning algorithms. The next section discusses the new architecture with more details.

3. A new ITS architecture based on collaborative planning agents

The new architecture (figure 2) is a layered one in which four planning agents operate. Each planning agent acts at one layer, but they collaborate in the planning process by exchanging information. The individualized course planner agent *(IC-Planner agent*) responsibility is to generate a course graph (plan) that fit to the current student group and according to a certain goal that is expressed as set of capabilities to be acquire by the student. The Lesson planner agent (*LE-Planner*) based on the course plan, dynamically create a plan of the lessons in the current session. As the learning actions are based on student abilities, the TA-Planner agent plans the sequence of activity the student have to do and choose a specific learning strategy to sustain the activity delivery. Then, the IP-Planner is in charge of the presentation of the activity by planning the sequence of basic action the student deals with. Any planning agent can communicate with the student through a learning interface. For instance, at the first layer or the architecture, the IC-Planner can interact with the student in order to establish the learning goals. In the next section, we give some details of the planning strategy in IC-Planner.

4. Using SIMPLAN as the foundation of planning processes in ITS

Due to space limitations, we give more details on only the IC-Planning agent, leaving the description of others to the extended version of this paper. Our architecture is open, so that different planning algorithms can be used provided they support *reactive plans*, that is, they can produce conditional plans that specify actions to be executed depending on the current situation.

For our initial implementation, we have decided to use a model-checking planning algorithm proposed by Kabanza et al. (Kabanza, Barbeau and St-Denis, 1997) and currently supported by the SIMPLAN planning system (Kabanza, 1999).

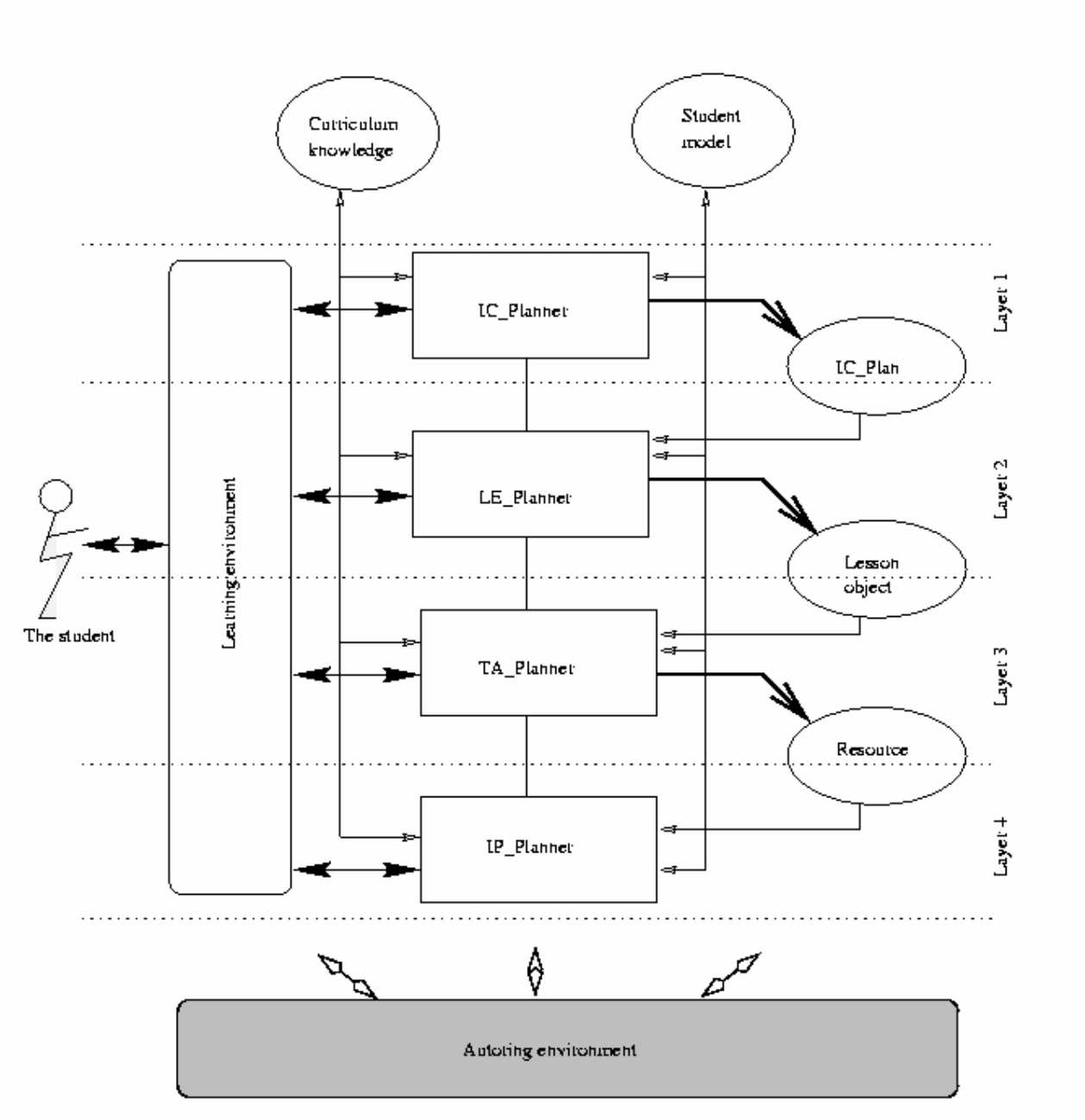

Figure 2: The new architecture

Reactive plans such those produced by SIMPLAN are finite state machine. They specify the actions to be executed by an agent depending on the current agent's state and current environment situation. This contrast with sequential plans often produced by other planning systems in which produced plans are just sequences of actions without any additional pre-condition or context under which actions in the plan are applicable. In fact, sequential plans are appropriate for predictable environments in which a planned sequence of actions is always

expected to provide the anticipated outcome. On the other hand, reactive plans such as those produced by SIMPLAN are suitable for unpredictable environments in which the agent has no control on events generated by the environment. Hence the agent cannot anticipate the outcome of its own actions and this is why a conditional plan that map agent actions to sets of expected situations is necessary. This is the case in ITS planning because the IC-Planning agent for example cannot anticipate choices made by the student when following a course.

Remember that our new architecture is built from Nkambou, Gauthier and Lefebvre's original one (Nkambou, Gauthier and Lefebvre, 1997). We have initially decided to keep parts of the original architecture in order to be able to make real-word tests as soon as possible by relying on the current implementation. In particular, the knowledge representation underlying the original architecture remains unchanged. What has changed so far is the number of agents, their communication and the planning algorithms they use. Thus like the original Nkambou, Gauthier and Lefebvre's architecture, our IC-Planning agent relies on *Curriculum Knowledge Transition Network (CKTN)* for representing planning knowledge. However, although SIMPLAN accepts different planning knowledge representations, none of them coincides with CKTN. The closest one is the *Action Description Language (ADL)* frequently encountered in the artificial intelligence planning field (Pednault, 1989). So we begin with a brief reminder of CKTN description. Then we explain how this knowledge is translated into an ADL representation acceptable for SIMPLAN. This should demonstrate that SIMPLAN effectively fits the new architecture, at least as far as IC-Planning is concerned. As for the other planning agents, the demonstration is made in an extended version of this paper as mentioned before.

4.1. IC-Planning: The actual process

By using CREAM modeling approach, a curriculum is represented as a structure that relate the subject matter in terms of capabilities (Gagné, Briggs & Wager 1992), instructional objectives, didactic methods, and pedagogical resources (learning materials). More specifically, the structure representing a curriculum is a network organization of capabilities, of instructional objectives defined on these capabilities and of pedagogical resources supporting the accomplishment of instructional objectives. The net contains links from instructional objectives to capabilities and this means that the achievement of instructional objectives contributes to the acquisition of capabilities. Those links are also labeled by pedagogical resources that support the achievement of instructional objectives through learning activities (exercises, demonstrations, problems, simulations...). The resulting network is called CKTN (Curriculum-knowledge-transition network).

The CKTN. In the CKTN the *input capabilities* of a transition are those that are prerequisites to the achievement of the objective involved in this transition. The *output capabilities* of a transition are those that are produced by the achievement of the objective involved in the transition. Thus, they produce two types of links between capabilities and transitions: *prerequisites* and *contribution* links. Figure 3 shows part of CKTN in the Baxter pump manipulation ITS (A tutor intended to nurses that teaches them how to carry out infusion task on a medical device called Baxter-Pump). A *prerequisite* link from a capability C to transition T expresses the fact that C is a precondition to the activation of T. A pre-requisite link is characterized by its nature (*mandatory*, *optional*) and by the minimum mastery level required on the source capability to be able to consider it sufficient for overstepping the link (*entry level)* and thus be able to eventually traverse the transition. This entry level is specified by using

integer values of an evaluation set. In this example, the evaluation set contains 4 values (1, 2, 3 and 4). According to Klausmeier's classification of concepts evaluation (Klausmeier, 1990), mastery levels 1,2,3 and 4 correspond respectively to "identify", "recognize", "classify" and "generalize" the concept. For instance, in figure 4, before being able to learn how to program an infusion rate, the nurse shall master the concept of infusion rate at level 1 (this means that she shall be able to identified the concept of infusion rate). In general, evaluation vocabularies are denoted in our system by an ordered set of integers representing different levels of acquisition. Since several vocabularies exists for describing the same type of capability, the designer must choose a vocabulary before proceeding to the construction. A *contribution* link qualifies the way in which the realization of a transition contributes to the acquisition of a capability: The contribution weight is specified on the link and is qualified as *weak*, *moderate* or *strong* contribution. Integer values indicate the expected mastery level of the concerning knowledge. Several transition nodes can contribute to the acquisition of one capability.

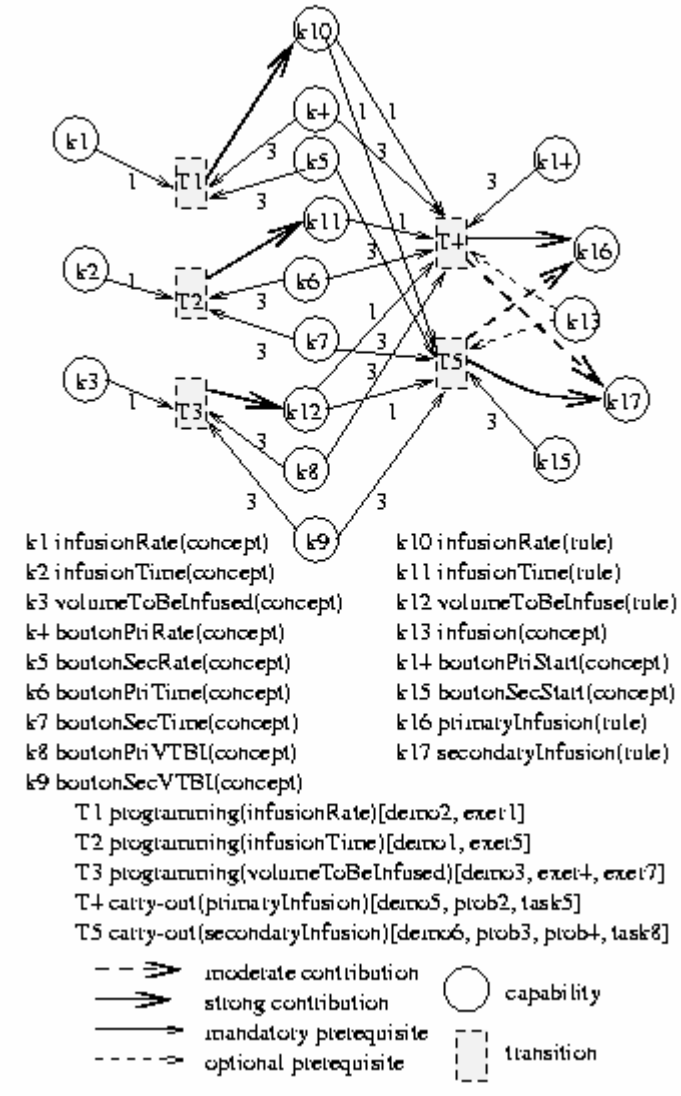

Figure 3. Part of CKTN on the Baxter Pump

CKTN is central in the old planning process in ITS. The core of the system reasoning during planning process is based on this network.

The Target group knowledge. We define a target group (TG) as a group of students' state of knowledge of various capabilities which may be part of several subject matters (curriculums). For instance, the knowledge of a novice nurse on the handling of the Baxter pump will not be the same as that of an advanced nurse. Therefore, a course on this topic should not include the same transitions for the former as for the latter. The advanced nurse would waste her time learning things that she already knows. Thus, these two groups of nurses constitute two different student target groups and the planner should build a course well-suited to each one.

The Training requirement. Training requirement is expressed either as a set of objectives that the course should reach, or as set of capabilities to be acquired by the student. It is also possible to mix these two approaches.

The old IC-planning process. The generation is performed by going through the CKTN. But before that, the TG state of knowledge is assigned to the capability nodes and the links which constitute the CKTN. The resulting graph is called a dynamic CKTN (DynCKTN) and we call this operation the marking of CKTN. More precisely, it consists in attributing

- to each prerequisite link a value in the list: {*acquired, partially acquired, not acquired*} indicating whether the minimum acquisition level on this link has been reached according to the target group state of knowledge,
- and to each capability a value in {*possessed, partially possessed, not possessed*} representing the acquisition standard of our target public on this knowledge and calculated from the levels assigned to the links.

After the marking process, the generation algorithm traverses the resulting DynCKTN to determine which transition have to be included in the course in order to permit the acquisition of the knowledge specified in the training requirements (specified for instance in terms of capabilities). To do this, we perform from each capability in training requirement, a backward chaining traversal of the sub-graph rooted at the capability in order to choose the transition judged as necessary for the acquisition of that capability. We first evaluate the immediate transition that contribute to the capability and then, since some transitions possess mandatory prerequisite knowledge which in turn has contributing transition, we have to trace the sub-graph back until we reach a transition without any pre-requisite or a capability already mastered by the student (as specified in the training requirement or seen by the marking). The choice of relevant transition is carried out by applying heuristic rules introduced into the system and which consider several parameters: the links between capabilities and transitions (prerequisite or contribution), the knowledge in the training requirement and also the DynCKTN. As the output of this process, a course graph is generated. Figure 4 shows the DynCKTN derived from the CKTN in figure 3 after the marking process. If the teaching goal is to make student acquire the concept k16 at a medium level, then one possible generated course graph will includes transitions T1, T3 and T4.

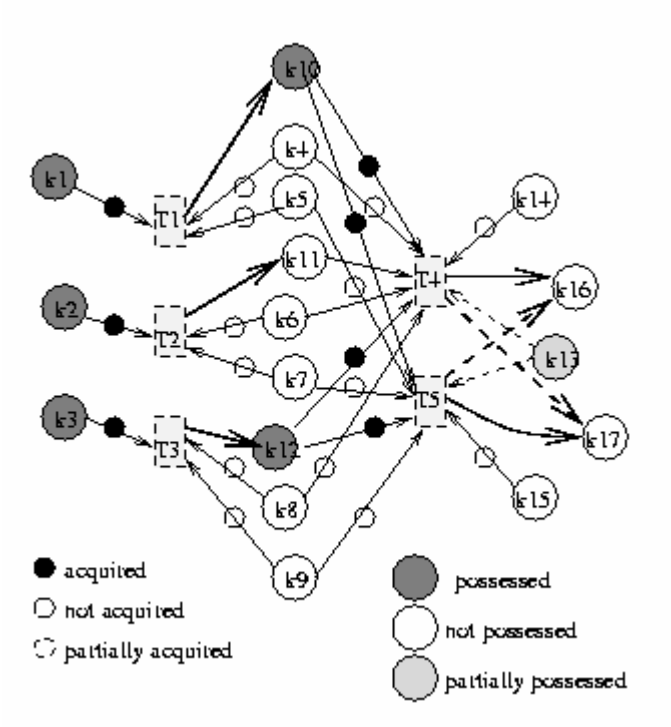

Figure 4: DynCKTN from the CKTN in figure 3

4.2. From CKTN to ADL

ADL specifies an agent's action in terms of *precondition*, *a set of removed facts*, and *a set of added facts*. Roughly, the *precondition* is a logical formula expressing facts that must be true in a state for the action to be meaningful in that state. In other words, if the precondition is true, then we are in a state where the agent *can* accomplish that action. Whether the agent *will* indeed accomplish the action or not, this will depend on whether or on the agent's plan specifies that action for that state. In fact, in every state, there may be several actions that the agent can execute. However, an agent can execute only one action at a time, so it must decide upon which action to execute. This is why it needs a plan that tells which action to execute in the current state. The *set of removed facts* specifies facts that are true in the current state become false after the action is executed, while the *set of added facts* specifies facts that are not true in the current state but becomes true after the execution of the action. Both the set of removed and added facts form the *effects* or *postconditions* of the action. Hence, given a description of a current state in term of facts true and an ADL description of an agent's action, a planning system can predict the state that would result from the execution of the action by removing facts in the set of removed facts and then adding facts in the set of added facts.

ADL is useful for describing complex actions through the use of variables and instantiation. That is, instead of specifying individual agent's actions, one can specify schemas of actions that share the same structure of precondition and effects. Moreover, ADL allows the specification of conditional effects in which facts are removed or added depending on the current state.

The translation of CKTN into ADL is quite easy. Input capabilities with their preconditions become represented by preconditions, transitions are replaced actions, while output capabilities are specified as added and removed facts. Figure 5 shows the ADL translation of the CKTN in Figure 3. This translation was made automatically, but the translation algorithm is omitted due to space limitation. It will be included in the forthcoming extended version of this paper.

```
(add-adl-op
     :name '(T1)
     :pre (adl-pre :var-gens '((?x1) (k1 ?x1) (?x4) (k4 ?x4) (?x5) (k5 ?x5))
     :form  '(and (>= ?x1 1) (>= ?x4 3) (>= ?x5 3)))
     :add (adl-add (adl-cond :form '(not (exists (?x) (k10 ?x) (>= ?x 10)))
     :lit '(k10 10)))
     :del (adl-del  (adl-cond :var-gens '((?x) (k10 ?x))
     :form '(< ?x 10)
     :lit '(k10 ?x))))
(add-adl-op
     :name '(T2)
     :pre (adl-pre :var-gens '((?x2) (k2 ?x2) (?x6) (k6 ?x6) (?x7) (k7 ?x7))
     :form  '(and (>= ?x2 1) (>= ?x6 3) (>= ?x7 3)))
     :add (adl-add (adl-cond :form '(not (exists (?x) (k11 ?x) (>= ?x 10)))
     :lit '(k11 10)))
     :del (adl-del  (adl-cond :var-gens '((?x) (k11 ?x))
     :form '(< ?x 11)
     :lit '(k11 ?x))))
(add-adl-op
     :name '(T3)
     :pre (adl-pre :var-gens '((?x3) (k3 ?x3) (?x8) (k8 ?x8) (?x9) (k9 ?x9))
     :form  '(and (>= ?x3 1) (>= ?x8 3) (>= ?x9 3)))
     :add (adl-add (adl-cond :form '(not (exists (?x) (k12 ?x) (>= ?x 7)))
     :lit '(k12 7)))
     :del (adl-del  (adl-cond :var-gens '((?x) (k12 ?x))
     :form '(< ?x 7)
     :lit '(k12 ?x))))
(add-adl-op
     :name '(T4)
     :pre (adl-pre :var-gens '((?x4) (k4 ?x4) (?x6) (k6 ?x6) (?x8) (k8 ?x8)
                               (?x11) (k11 ?x11) (?x10) (k10 ?x10) (?x12)
                               (k12 ?x12) (?x14) (k14 ?x14))
     :form  '(and (>= ?x4 3) (>= ?x8 3) (>= ?x6 3) (>= ?x11 1) (>= ?x10 1)
                  (>= ?x12 1) (>= ?x14 3)))
     :add (adl-add (adl-cond :form '(not (exists (?x) (k16 ?x) (>= ?x 8)))
     :lit '(k16 8))(adl-cond :form '(not (exists (?x) (k17 ?x) (>= ?x 5)))
     :lit '(k17 5)))
     :del (adl-del  (adl-cond :var-gens '((?x) (k16 ?x))
     :form '(< ?x 8)
     :lit '(k16 ?x)) (adl-cond :var-gens '((?x) (k17 ?x))
     :form '(< ?x 5)
     :lit '(k17 ?x))))
(add-adl-op
     :name '(T5)
     :pre (adl-pre :var-gens '((?x5) (k5 ?x5) (?x7) (k7 ?x7) (?x9) (k9 ?x9)
                           (?x10) (k10 ?x10) (?x12) (k12 ?x12) (?x15) (k15 ?x15))
     :form  '(and (>= ?x5 3) (>= ?x7 3) (>= ?x9 3) (>= ?x10 1)
                  (>= ?x12 1) (>= ?x15 3)))
     :add (adl-add (adl-cond :form '(not (exists (?x) (k17 ?x) (>= ?x 10)))
     :lit '(k17 10))(adl-cond :form '(not (exists (?x) (k16 ?x) (>= ?x 5)))
     :lit '(k16 5)))
     :del (adl-del  (adl-cond :var-gens '((?x) (k17 ?x))
     :form '(< ?x 10)
     :lit '(k17 ?x)) (adl-cond :var-gens '((?x) (k16 ?x))
     :form '(< ?x 5)
     :lit '(k16 ?x))))
```

Figure 5 : ADL corresponding to the CKTN in figure 3.

Because of space limitations, we cannot give a full description of the SIMPLAN's implementation of ADL. Accordingly, we do not expect the reader to fully understand the above

example without a previous reading of the SIMPLAN's manual (Kabanza, 1999). Nonetheless, we provide a rough explanation of the syntax in the figure which allows a coarse understanding. Roughly, SIMPLAN uses a Lisp-like notation (that is, prefix notation with parenthesis to separate different elements of an expression). Each action schema is described by an *adl-add-op* expression, with the keywords *:pre*, *:add* and *:del* for, respectively, the precondition, added facts and deleted facts. In addition we have a *:name* keyword that uniquely identifies each action schema. For instance, the first *add-adl-op* expression describes the transition *T1* in the CKTN of Figure 3. The expression *:var-gens* is roughly a declaration of the variables involved in description of the action together with some information on how the variable will get instantiated. The *:form* keyword describes a logical formula that must hold for the precondition to be true, for a literal to be deleted (in the *:add* component) or to be deleted (in the *:delete* component).

4.3 SIMPLAN Planning with CKTN

Once we have a translation of CKTN into ADL it becomes easy to generate plans. The target is easily expressed as a *goal* for SIMPLAN while the student's profile corresponds to *the initial state* for SIMPLAN. Figure 6 shows a trace generated by SIMPLAN from a goal and initial state corresponding respectively to the target knowledge and student's profile highlighted in the CKTN of Figure 3, and with the ADL actions of Figure 5 which are translated from the CKTN of Figure 3. The plan is at the end of the trace.

```
SimPlan 1.0

Init state: ((K15 5) (K14 5) (K9 4) (K8 4) (K7 4) (K6 4) (K5 3) (K4 3)
             (K3 2) (K2 1) (K1 2))

Goal: (EVENTUALLY (K16 5))

Reactive Plan:

[ID 0 state ((k15 5) (k14 5) (k9 4) (k8 4) (k7 4) (k6 4) (k5 3) (k4 3)
             (k3 2) (k2 1) (k1 2)) ACTION ((T3) 1)]
[ID 1 state ((k15 5) (k14 5) (k12 7) (k9 4) (k8 4) (k7 4) (k6 4) (k5 3)
             (k4 3) (k3 2) (k2 1) (k1 2)) ACTION ((T2) 2)]
[ID 2 state ((k15 5) (k14 5) (k12 7) (k11 10) (k9 4) (k8 4) (k7 4)
             (k6 4) (k5 3) (k4 3) (k3 2) (k2 1) (k1 2)) ACTION ((T1) 3)]
[ID 3 state ((k15 5) (k14 5) (k12 7) (k11 10) (k10 10) (k9 4) (k8 4)
             (k7 4) (k6 4) (k5 3) (k4 3) (k3 2) (k2 1)
             (k1 2)) ACTION ((T5) 4)]
[ID 4 state ((k17 10) (k16 5) (k15 5) (k14 5) (k12 7) (k11 10) (k10 10)
             (k9 4) (k8 4) (k7 4) (k6 4) (k5 3) (k4 3) (k3 2) (k2 1)
             (k1 2)) ACTION ((T5) 5)]
[ID 5 state ((k17 10) (k16 5) (k15 5) (k14 5) (k12 7) (k11 10) (k10 10)
             (k9 4) (k8 4) (k7 4) (k6 4) (k5 3) (k4 3) (k3 2) (k2 1)
             (k1 2)) ACTION ((w 1) 5)]
```

Figure 6. SIMPLAN generated plan from ADL of figure 5.

More explanations on the representation of plans by SIMPLAN are needed in order to better understand the plan in Figure 6. A plan is an automaton with states identified by numbers (ID) and labeled with sets of propositions true. To each state corresponds the action to be executed by the agent and a set of expected resulting states. More specifically, because of nondeterminism

resulting from uncontrollable events, it may be the case the outcome of an action is uncertain. In this case, we have a set of successor states, which means that when the action is effectively executed, we cannot predict beforehand the outcome, but the resulting state is expected to be one among the set of successors. For instance, the first two lines of the reactive plan in Figure 6 specify that state with ID 0 has the attached set of proposition true. Whenever these conditions hold, the agent must perform transition T1 and the expected resulting state is the with ID 1. In this particular example, all actions are deterministic so that we always have only one successor state.

5. Conclusion

We have presented a new multi-agent ITS architecture that includes planning agents. In this architecture, planning agents collaborate in order to make the learning process more adapted to the student. The planning process is supported by SIMPLAN, a recently developed planning system that combines model checking and Markov decision theoretic techniques. Reactive plans produced by SIMPLAN are finite state machine that specify the actions to be executed by an agent depending on the current agent's state and current environment situation (student knowledge, learning goals, …). By considering teaching/learning process in an ITS as a multi-layer planning task, we lead to a uniform view of ITS processes from one layer to another. What change is the semantic of the planning process. Thus communication within the ITS between planning agent is facilitated. Also, generic planning agent view of ITS components make it easy to re-use the same technology in many application domain. This represent a major contribution in component-based ITS which is the actual focus of several ITS researches (Brusilovsky, 1995; Ritter and Koedinger, 1996; Ritter, Brusilovski and Medvedeva, 1998; Devedzic, Radovic, and Jerinic, 1998). Our future works will concentrate on the collaboration between planning agents and more experimentation in real word applications. Also, we will easier the authoring process of planning agent by developing tools dedicated to non programmers users.

6. References

André, E. and Rist, T. Coping with temporal constraints in multimedia presentation planning.In *Proc. of 13th National Conference on Artificial Intelligence*, pages 142--147. AAAI Press, 1996.

Bacchus, F. and Kabanza, F. Using temporal logic to control search in a forward chaining planner. In M.~Ghallab and A.~Milani, editors, *New Directions in AI Planning*, pages 141--153. ISO Press, Amsterdam, 1996.

Barbeau, M., Kabanza, F. and St-Denis, R. A method for the synthesis of controllers to handle safety, liveness, and real-time constraints. *IEEE Transactions on Automatic Control*, 1998.

Bass, E.J. Towards an Intelligent Tutoring System for Situation Awarwness Training in Complex, Dynamic Environments. In proceedings of the 4th International Conference on ITS., pp..26-35. Springer-Verlag, Berlin, 1998.

Brecht, B. Determining the focus of instruction: Content planning fir Intelligent Tutoring Systems. Research report 90-5. Department of Computer Science, University of Saskatchewan, 1990.

Brusilovsky, P. Intelligent learning environments for programming: The case for integration and adaptation. In: Proceedings of the AIED'95, pp.1-8, AACE, 1995.

Burn and Caps. *Intelligent Tutoring Systems*. Lawrence Erlbaum Associates, Hillsdale, NJ., 1988.
Choquet, C., Danna, F., Tchounikine, P. and Trichet, F. Modelling the knowledge-Based Components of a Learning Environment within the Task/Method Paradigm. In proceedings of the 4th International Conference on ITS., pp..56-65. Springer-Verlag, Berlin, 1998.
Dean, T., Kaelbling L.P., Kerman, J. and Nicholson, A. Planning under time constraints in stochastic domains. *Artificial Intelligence*, 76(1--2):35--74, 1995. Elsevier.
Devedzic, V., Radovic, D. and Jerinic, L. On the notion of components for intelligent tutoring systems. In: Proceedings of the 4th International Conference on ITS., pp..504-513. Springer-Verlag, Berlin, 1998.
Frasson, C., Mengelle, T., Aïmeur, E. and Gouardères, G. An Actor-Based Archtecture for Intelligent Tutoring Systems. In proceedings of the 3rd International Conference on ITS., pp 57-65. Springer-Verlag, Berlin, 1996.
Gagné, R.M., Briggs, L.J. & Wager, W. *Principles of Instructional Design* (4th edition). Orlando, FL: Harcourt Brace Jovanovich, 1992.
Kabanza, F. SimPlan's Manual. 1999. The manual is part of the system which can be downloade from the site http://www.dmi.usherb.ca/~kabanza/simplan.
Kabanza, F., Barbeau, M. and St-Denis, R. Planning control rules for reactive agents. *Artificial Intelligence*, 5(1):67--113, 1997. Elsevier.
Klausmeier, H.J. Conceptualizing. In: B.F. Jones and L. Idol (Eds). Dimensions of thinking and cognitive instruction, pp. 93-138. NJ:LEA, 1990.
Le, T.H. Planification de l'ensewignement individualisé dans un système tutoriel intelligent à grande échelle. Thèse de doctorat, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1998.
Nkambou, R. et Gauthier, G. Integrating WWW resources in an intelligent tutoring system. *Journal of Network and Computer Science Application*, vol. 19, no. 4. Academic Press, London, 1996.
Nkambou, R., Gauthier, G. and Frasson, C. "Un modèle de représentation des connaissances relatives au contenu dans un système tutoriel intelligent". Science et Techniques Éducatives. 31 (1), 299-330. 1997.
Nkambou, R., Gauthier, G. et Lefebvre, B.. Un modèle d'architecture de STI pour l'enseignement à grande échelle. *Dans Environnement interactif d'apprentissage avec ordinateur*, pp. 236-249. HERMES, Paris, 1997.
Nkambou, R., Lefebvre, B. and Gauthier, G. 1996. "A Curriculum-Based Student Modelling for intelligent tutoring systems". *Proceedings of the fifth International Conference on User Modelling*, pp. 86-96, Kailua-Kona, Hawaii
Pednault, E.P.D. Exploring the Middle Ground between STRIPS and the Situation Calculus. In : Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR' 89), pp. 324-332, 1989.
Ritter, S. and Koedinger, K.R. An architecture for plug-in tutor agents. Journal of Artificial Intelligence in Education, 7, 315-347. 1996.
Ritter, S., Brusilovsky, P. and Medvedeva, O. Creating more versatile intelligent. Learning environments with a component-based architecture. In: Proceedings of the 4th International Conference on ITS., pp. 554-563. Springer-Verlag, Berlin, 1998.
S.J. Russel and P. Norvig . Artificial Intelligence, A Modern Approach. Prentice Hall, 1995.
Sycara, K.P. et al. Intelligent Agents. *AI Magazine*, 19 (2): 11-92. 1998.

Biography

Roger Nkambou received a Ph.D. (1996) in computer science from the University of Montreal. Dr. Nkambou is currently an assistant professor at the department of mathematics and computer science of the University of Sherbrooke. His research interests include ITS authoring systems, Distributed Multimedia Resources in ITS, and Distributed ITS. Roger Nkambou is a regular member of the GRITI (Inter-University Group on Tutoring System).

Froduald Kabanza was awarded a Ph.D. (1992) in computer science from the University of Liege (Belgium). Dr Kabanza is currently an associated professor at the department of mathematics and computer science of the University of Sherbrooke. His research interests include planning, decision making, machine learning and verification in intelligent systems. He is also a regular member of the GRITI.