UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCES AND ENGINEERING

KNOWLEDGE REPRESENTATION FOR AFFECTINE AND ADAPTIVE TUTORING SYSTEMS

by

ACHILLEFS - NIKOLAOS DOUGALIS

PhD Dissertation
Presented
in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

Heraklion, July 2023

UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE

Knowledge Representation for Affective and Adaptive tutoring systems

PhD Dissertation Presented
by **Achillefs Nikolaos Dougalis**
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

APPROVED BY:

Author: Achillefs-Nikolaos Dougalis

Supervisor: Dimitris Plexousakis, Professor, University of Crete

Committee Member: Panagiotis Anastasiadis, Professor, University of Crete

Committee Member: Antonis Argyros, Professor, University of Crete

Committee Member: Michail Kalogiannakis, Associate Professor, University of Crete

CommitteeMember:Theodore Patkos, Principal Researcher, Institute of Computer Science, FORTH

Committee Member: Constantine Stephanidis, Professor, University of Crete

Committee Member: Ioannis Tsamardinos, Professor, University of Crete

Department Chairman: Antonis Argyros

Heraklion, 11 July 2023

To Marianna

Acknowledgments

I would like to thank the committee members for their help, support and patience, especially my supervisor, Professor Dimitris Pleksousakis, as well as Dr Patkos for their guidance.

I would also like to thank the members of the Information Systems laboratory, and Human Computer Interaction laboratory for their help and support of this project, as well as all the volunteers that trained and evaluated Afflog and Afflog RL.

Abstract

In recent years, there has been a meteoric rise in the use of educational technology. Intelligent Tutoring Systems (ITS) and Affective Tutoring Systems (ATS) are computer systems that can successfully teach courses to users at all levels of education, using AI techniques (the former) and Affect Sensing technologies (the latter) in order to facilitate learning. These systems are typically built around a specific subject such as algebra or genetics,etc, making reusability in different domains difficult.

In contrast to ITS, Adaptive Learning Systems are more domain-agnostic, being able to teach more than one course. Adaptive Learning Systems concentrate on the presentation of a course by adapting it to the user's learning preferences. These adaptations are based on learning theories that classify learners according to the way they learn; these theories are known as Learning Styles. Two major disadvantages of these systems is that they do not offer any type of feedback to the user (emotional or cognitive), and that there are doubts in the scientific community concerning the validity of Learning Styles Theories due to lack of relevant scientific evidence.

Massive Online Open Courses (MOOCs) are the latest word in educational technologies; they deal with the problem of integrating many courses, users and instructors in a single platform. In order to be able to build better MOOC platforms, a course-teaching system should combine the advantages of ITS, ATS and Adaptive Learning Systems. Also, there is a need for a unified model that will be able to adequately represent a range of different courses. Finally this system should have the ability to calibrate its teaching strategies during interaction with users.

This doctoral dissertation describes the design, implementation and evaluation of AffLog (affective Logic) Tutor, an ATS with the following specifications: The domain, tutoring and student models are designed using the Predicate Calculus and the Event Calculus. They are then implemented following the Answer Set Programming (ASP) formalism using the programming language Clingo. Also, the student model contains information about the user's learning style according to the Felder-Silverman model. AffLog uses AI methods such as Planning and Projection in order to select the most suitable parts of the course for the current user according to the user's learning style and suitably present these parts to the user. The affective model of the system is designed to react to the user's current emotional state providing advice and encouragement, thus facilitating the learning process.

In order to evaluate the system, a simple course containing instructions on how to play the "Settlers of Catan" board game was designed and implemented. Evaluation of the system showed that users had high learning gains from their interaction with the system.

Also, in an effort to find if Learning Styles are relevant to AffLog's learning gains, a second ATS system, AffLogRL was developed. AffLogRL is similar to AffLog except that the latter replaces all modules that make use of Learning Styles with a Reinforcement learning agent. This agent uses the experience it accumulates by interacting with the users in order to formulate a teaching policy. After the training of AffLogRL with over 100 human users, the system was evaluated. The results showed that AffLogRL performed similarly to the AffLog system indicating that with more training and using Relational approaches the RL will perform better.

Keywords: Affective Tutoring Systems, Adaptive Learning Systems, Answer Set Programming, Learning styles, Reinforcement Learning.

Supervisor: Dimitris Plexousakis
Professor
Computer Science Department
University of Crete

Περίληψη

Τα τελευταία χρόνια, έχει παρατηρηθεί δραματική αύξηση στην χρήση τεχνολογιών εκπαίδευσης. Ιδιαίτερο ενδιαφέρον παρουσιάζουν τα Ευφυή Συστήματα Διδασκαλίας (Intelligent Tutoring Systems-ITS). Τα ITS είναι συστήματα Τεχνητής Νοημοσύνης που μπορούν να διδάξουν χρήστες από όλα τα επίπεδα εκπαίδευσης, αντικαθιστώντας τον διδάσκοντα. Μια υποκατηγορία των ITS, είναι τα Συναισθηματικά Συστήματα Διδασκαλίας (Affective Tutoring Systems-ATS), τα οποία χρησιμοποιούν τεχνικές ανίχνευσης συναισθημάτων του χρήστη προκειμένου να διευκολύνουν την διδασκαλία. Τα ITS και ATS, συνήθως σχεδιάζονται ώστε να διδάξουν συγκεκριμένα μαθήματα όπως άλγεβρα , γενετική και προγραμματισμό. Αυτό καθιστά δύσκολη την επαναχρησιμοποίηση αυτών των συστημάτων για διαφορετικά πεδία μάθησης.

Σε αντίθεση με τα ITS, τα Συστήματα Προσαρμοζόμενης Εκμάθησης (Adaptive Learning Systems) έχουν σχεδιαστεί για πιο γενική χρήση, επιτρέποντας την παρακολούθηση διαφορετικών μαθημάτων από τον χρήστη μέσω του ίδιου συστήματος. Αυτά τα συστήματα επικεντρώνονται στο να προσαρμόσουν την μορφή της παρουσίασης του μαθήματος ανάλογα με τις μαθησιακές προτιμήσεις του χρήστη. Οι προσαρμογές αυτές, βασίζονται σε θεωρίες μάθησης οι οποίες προσπαθούν να κατηγοριοποιήσουν τους εκπαιδευόμενους αναλόγως με τον τρόπο που μαθαίνουν, και λέγονται Μέθοδοι Μάθησης (Learning Styles). Τα συστήματα αυτά έχουν δύο μειονεκτήματα. Πρώτον, δεν προσφέρουν ανατροφοδότηση στον χρήστη, και δεύτερον υπάρχουν αμφιβολίες στην επιστημονική κοινότητα σχετικά με την εγκυρότητα των Μεθόδων Μάθησης.

Τα Μαζικά ανοιχτά διαδικτυακά μαθήματα (MOOCs) είναι η τελευταία λέξη στις τεχνολογίες εκπαίδευσης. Και καταπιάνονται με το πρόβλημα της ενσωμάτωσης μαθημάτων, χρηστών και εκπαιδευτών στην ίδια πλατφόρμα. Για την σχεδίαση καλύτερων MOOCs , ένα σύστημα διδασκαλίας χρειάζεται να συνδυάζει τα πλεονεκτήματα των συστημάτων ATS, ITS και Προσαρμοζόμενης Εκμάθησης. Επιπλέον, ένα τέτοιο σύστημα χρειάζεται ένα ενοποιημένο Μοντέλο το οποίο θα μπορεί να αναπαριστά ένα εύρος διαφορετικών μαθημάτων και χρηστών, χωρίς να είναι σχεδιασμένο πάνω σε ένα μάθημα.

Η παρούσα διατριβή περιγράφει τον σχεδιασμό, την υλοποίηση και την αξιολόγηση ενός συστήματος ATS, του Afflog με τις εξής ιδιότητες: Τα μοντέλα πεδίου, διδασκαλίας και χρήστη (domain model, tutoring model και student model) του συστήματος σχεδιασμένα με λογικά κατηγορήματα χρησιμοποιώντας τον κατηγορηματικό Λογισμό, και την υλοποίηση τους στην προγραμματιστική λογική του Answer Set Programming και του Event Calculus χρησιμοποιώντας την γλώσσα προγραμματισμού Clingo. Επίσης, το μοντέλο χρήστη περιέχει και τις Μεθόδους Μάθησης του χρήστη σύμφωνα με το Felder-Silverman learning style model. Το σύστημα χρησιμοποιεί μεθόδους Τεχνητής Νοημοσύνης όπως Planning και Projection ώστε να επιλέξει τα μέρη του μαθήματος που είναι πιο κατάλληλα για τον χρήστη σύμφωνα με την Μέθοδο Μάθησής του, και να του τα παρουσιάσει. Το συναισθηματικό μοντέλο του συστήματος είναι σχεδιασμένο να αντιδρά στην παρούσα συναισθηματική κατάσταση του χρήστη παρέχοντας συμβουλές, βοηθώντας έτσι στην διδασκαλία.

Ένα απλό μάθημα με οδηγίες για το επιτραπέζιο παιχνίδι "Οι Άποικοι του Κατάν" σχεδιάστηκε και υλοποιήθηκε για την αξιολόγηση του συστήματος. Η αξιολόγηση έδειξε ότι οι χρήστες είχαν σημαντικά μαθησιακά οφέλη (learning gains).

Επίσης, ένα δεύτερο ATS σύστημα, το AfflogRL, σχεδιάστηκε και υλοποιήθηκε. Το AfflogRL αντικαθιστά τα μέλη του συστήματος που χρησιμοποιούν την Μέθοδο Μάθησης του χρήστη με έναν πράκτορα Μηχανικής Μάθησης (Machine Learning agent) ο οποίος χρησιμοποιεί Ενισχυτική

Μάθηση (Reinforcement Learning) για να μετατρέψει την εμπειρία από τις συνεδρίες του με χρήστες, σε διδακτική στρατηγική (learning policy). Μετά από την εκπαίδευση του συστήματος με πραγματικούς χρήστες, η αξιολόγηση της διδακτικής στρατηγικής των δύο συστημάτων έδειξε ότι τα μαθησιακά οφέλη των χρηστών και των δύο εφαρμογών είναι μεγάλα, με τα δύο συστήματα να αποδίδουν παρόμοιες τιμές με μικρή απόκλιση. καθιστώντας και το δεύτερο σύστημα ως κάτι χρήσιμο τουλάχιστον στο πλαίσιο των τεχνολογιών εκπαίδευσης.

Λέξεις κλειδιά: Συναισθηματικά Συστήματα Διδασκαλίας, Συστήματα Προσαρμοζόμενης Εκμάθησης, Answer Set Programming, Μέθοδοι Μάθησης, Ενισχυτική Μάθηση

Επόπτης: Δημήτρης Πλεξουσάκης
Καθηγητής
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

In recent years, there has been a meteoric rise in the use of educational technology. At the same time, we witnessed the adverse effects of the COVID-19 pandemic on education at all levels, as efforts to contain the virus closed schools and universities around the world leaving more than one billion students and educators to rely solely on e-learning technologies [69]. However, online education was hindered by poor infrastructure including network, power, inaccessibility, and unavailability issues. In addition, both educators and students did not have the sufficient skills and training to use these systems making their usage impossible regardless of infrastructure availability and leading many educators to "reinvent the wheel" from an educational perspective, adopting a variety of technologies that were not necessarily designed for this purpose such as microsoft teams [70], zoom [71], etc. Finally, the educational technology companies themselves were caught unprepared for such a crisis. All these factors resulted in poor user experience. Nevertheless, this experience can be used in order to inspire new educational models. Now is the time for education technology to develop robust strategies and hybrid systems in order to cope with the post-coronavirus era.

This doctoral dissertation describes a work of combining the advantages of different educational systems in order to create a single Artificial-Intelligence-driven course-teaching system. It is my belief that human to human teaching, including synchronous and asynchronous methods, and face to face or electronic modes will never be surpassed by teaching from AI-driven systems, nor should this be attempted. This is because, in my opinion, these systems should be used for simple instruction and revision, as it is not their place to create trust, mutual respect, the feeling of a shared experience, and even inspiration and a sense of belonging to students. However, the amount of knowledge and practical skills that may be required at present is considerable as well as tremendously diverse and there are simply not enough human instructors to cover all the teaching tasks that may be needed. That is the reason behind the use of coursebooks, user manuals, self-improvement books, instructional videos and DIY web pages. AI-driven educational systems are the next evolutionary step of these tools as they can (and as they are already doing) cover most if not all these tasks.

Intelligent Tutoring Systems (ITS) [72], are computer systems that can successfully teach course subjects to users from all levels of education using AI techniques in order to facilitate learning. These systems often present exercises and tests to the user in order to assess the knowledge that the user gained. They also provide feedback to the user by offering hints and explanations depending on the user's progress during system usage. An ITS typically consists of the *domain model*, that contains information about the course or courses that the ITS teaches, the *student model,* that contains all the information regarding the user available to the system, *the tutor model,* that handles the teaching process by utilizing information taken from the domain and student models, and finally, the perception module and the user interface, that are responsible for interacting with the

user.

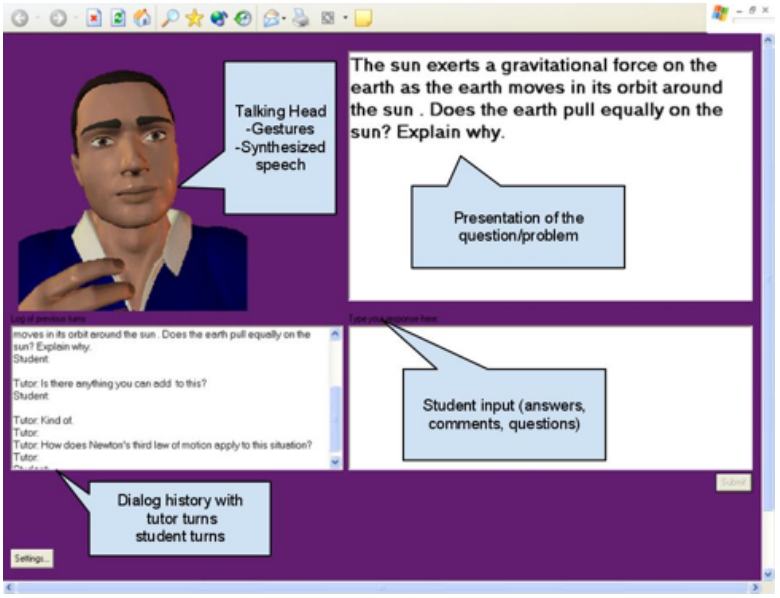

Figure 1.1 : an Intelligent Tutoring system [11]

A major disadvantage of these systems is that they are typically built around a specific subject such as algebra [73] or genetics [74], making reusability difficult as the tutoring model is heavily intertwined with the student and domain models. Another disadvantage is the plausibility problem [35]. That is, that in contrast to a human teacher, an ITS may not be "believable" or relatable enough for the user to pay the necessary attention to it thus impairing the teaching process.

Affective Tutoring Systems (ATS) [75], extend ITS by being able to detect the emotions of the user utilizing cameras and other kinds of sensors and then react to them accordingly as a human tutor would do. For example, such systems can offer words of encouragement to users who exhibit signs of confusion or help them by offering feedback according to their progress on the current course. These affective reactions make ATS relatable to the user thus solving the plausibility problem to a certain extent. Even with this problem solved however, both ITS and ATS are still quite far from the competence of a human teacher because, unlike teachers, these systems cannot adapt their teaching strategies according to a student's learning needs. This lack of adaptation inspired the design of the Adaptive Learning Systems [76].

Adaptive Learning Systems also known as E-Learning systems are usually more domain-agnostic as they concentrate on the presentation of a course to a specific user type rather than on the content of the course. The presentation of the course is done according to the user's learning preferences, including models that classify the user according to his/her preferred ways of learning, commonly known as Learning Styles [4]. For example, a user may prefer diagrams and videos rather than text or audio, or a course explaining the general idea first and then proceeding on the details rather than the opposite. A major disadvantage is that such systems do not offer any type of feedback to the user (emotional or cognitive). Also, because they do not take the content of a course into account, there is no way to assess the user's knowledge that was gained from the learning experience. Another problem is that to this day there has been a lack of methodologically sound studies of Learning Styles [4] and therefore there is no adequate scientific evidence to justify incorporating them in educational systems.

Massive Online Open Courses (MOOCs) [77] are the latest word in educational technology, dealing

with the problem of integrating many courses, users and instructors in a single platform in order to support university courses, community outreach programs, professional development, and corporate training applications [77]. In order to build better MOOC platforms a course teaching system should combine the advantages of ITS, ATS and Adaptive Learning Systems. Also, there is a need for a unified model that will be able to adequately represent a range of different courses. Finally this system should have the ability to calibrate its teaching strategies during interactions with users. The design and implementation of such a system introduces the following research aims/challenges:

1. The structure of the course should be quite flexible in order to be able to represent as many varieties of courses as possible, including school courses, university courses, and tutorials for a task.

2. The architecture of such a system should be modular in order to allow it or a subset of its components to be successfully used by MOOC platforms, and other educational systems. This would also help on future updates of the system.

3. The system should be able to deal with uncertainty issues regarding the user. Specifically the system needs to be able to monitor the user during his/her interaction with the system in order to ensure that the user comprehends all parts of the course and also to be able to successfully react to the user's emotions.

4. As ATSs are built with specific emotions and emotional teaching strategies according to a specific domain, a domain-agnostic system should deal with a variety of emotions that correspond to the mentality of a student, as well as a straightforward strategy to monitor these emotions.

5. While the detection of general emotions is already a challenging Computer Vision task, detection of emotional cues related to learning, such as flow or boredom, is even more challenging due to the absence of relevant datasets. By this we mean datasets dedicated to specific emotions associated with learning such as "flow", "boredom", "confusion, etc.

6. Although ITSs do not require human tutors to complement them, a more generalized system will require participation and decision making from an actual tutor.

7. Course generation, and the integration of existing courses in the system should be a fast and straightforward task.

8. Such a system should be able to incorporate the user's prior knowledge gained from previous courses in order to avoid teaching the same concepts multiple times.

9. Since there are many learning style theories and metrics a suitable theory should be used to represent a user better.

10. Adaptation and fine tuning of the teaching strategies of the system can offer many challenges, due to the large search space that the adaptive systems have to deal with, and due to the complexity of teaching in general.

11. Learning Styles may not be an adequate way to represent the user's learning ability anymore. New evidence suggests that Learning Styles may be a myth, mainly due to lack of scientific data supporting their evidence.

In this work, we focus on challenges 1 to 4 and 9 to 11. Although course generation and the utilization of the prior knowledge of the user are important, we believe that these challenges should

be addressed in a future study, in which we could generate a number of courses and study the experience of a user with each of these courses. Likewise, emotion detection and human tutor integration will not be addressed in this work, as they require studies in different scientific fields, specifically Computer Vision and pedagogical studies.

In order to solve the rest of the presented challenges, the following research hypotheses are made:

1. Knowledge Representation (KR) [62] methods such as the logical paradigm of Answer Set Programming (ASP) [2] and the Event Calculus (EC) action language [3] can be used in order to create a formal model and implementation for a Tutoring System including representation of the user, the user's emotional state as well as the tutoring course, while also dealing with uncertainty issues.

2. Such an approach also enables the use of powerful AI techniques such as Planning [62] and Projection [62] that can be used in order to solve problems such as the correct ordering of the parts of the course and the resolution of the actions of the tutor and the user .

3. If the Event Calculus can model all the possible states of the system and the user at any given time during a teaching session, as well as model the actions that the system performs, then it is feasible to model the teaching session as a Markov Decision Process (MDP) [63].

4. A Reinforcement Learning (RL) agent [64] that makes use of the MDP model from (2) is an adequate tool to optimize the system's decision making process and thus utilize the best teaching strategies.

5. The Felder-Silverman learning style model [1] as well as the emotions and affective strategies that are proposed in this work are suitable to be used for the adaptive and affective parts of the proposed system. The Felder-Silverman learning style model is the most common model used by the Adaptive Learning systems [65] as the learning dimensions it describes can be easily implemented in computer programs. For the Affective module, we used the six emotions identified by Craig et al [30] and developed the affective strategies based on a number of different ATS.

6. Due to the lack of scientific evidence for the effectiveness of Learning Style models the Felder-Silverman model may not be suitable for adaptation purposes. In this case, machine learning adaptation, such as Reinforcement Learning, could mitigate that problem and achieve better learning gains.

To validate the above hypotheses the following tasks were performed:

- The design, and implementation of the Affective Logic tutor (AffLog), an ATS with the following features:
    - A domain model that uses logical predicates to describe a course written in the Answer Set Programming formal language and executed by the Clingo reasoning system [5] following the stable models semantics.
    - The design of this model was as general as possible in order to be able to support a variety of different courses. The model also includes tests to verify the knowledge of the user.
    - A simple course that will be taught by the system as a use case. We created a course that teaches the user how to play the Settlers of Catan [66] board game.

- A student model containing the user's learning style preferences according to the Felder-Silverman learning style model [1].

- A Tutoring model that uses the Discrete Event Calculus [6] to plan all the system's possible actions. That is, to formulate all the different courses from the parts available, and also update the states of the system and the user after each action. We also implemented a course selection method that selects a course suited to the user's learning styles.

- A reactive affective model that will offer emotional response according to the user's current affective state which consists of the user's current emotion and the intensity of said emotion.

- We created AffLog RL, an extension of AffLog, by replacing the planning and selection methods of AffLog with a Reinforcement Learning policy.

  - Answer Set Programming and Event-Calculus-driven planning similar to the Afflog system was used for creating the MDP state space.

  - Event Calculus state update is similar to AffLog, but in this case was also used for constraining the action space from a given state, by rejecting the actions that are impossible from this state. Thus helping with the agent's action selection algorithm.

  - Also, Learning Styles were not used and the resulting Reinforcement Learning policy replaced the reactive affective model by incorporating emotional response actions.

  - We used the Q-learning algorithm [67] with a soft-max action selection strategy.

  - The RL system was trained by over 100 human users, each in 35-minute sessions in order to find the course that better fits all the users.

  - We created a reward function that combines the progress of the user through the chosen course, as well as the user's emotional state. This ensures that the system will eventually select parts of a course that influence users positively and also favor the more comprehensive parts.

- Comparison of the evaluation results of the two systems.

  - Each system was evaluated by twenty users. Evaluation included a pretest exam, a 35-minute session with the system and a post test exam. The users also had to fill a UX questionnaire [68] to describe their experience with the system.

  - Using the difference of the scores of the exams, we calculated the learning gains [8] of the users. Users of both systems achieved high learning gains of similar value. We believe that given the simplicity of the RL methods used, as well as the low training sample used for AffLog RL, gives AffLog RL the potential to further surpass the AffLog system, thus eliminating the need for the use of Learning Styles in a tutoring system.

The research contribution of this work is that to our knowledge, this is the first ATS system that

utilizes Knowledge Representation (KR) methods, and also attempts to create a course teaching system that combines the advantages of ITS, ATS and Adaptive Learning Systems. Specifically, the planning, projection, and selection methods demonstrate how known KR methods can be used in a new domain such as - but not restricted to - teaching.

The ASP formalism used by this work also has advantages over other types of programming languages. ASP is a simple and intuitive modeling language that comes with software for processing programs and there is no need to learn a coding language in order to use it. ASP has a number of advantages over other AI programming paradigms. First, ASP is a declarative programming language, meaning that programs are written in terms of what is to be achieved, rather than how it is to be achieved. This makes ASP programs more concise and easier to read and understand. Second, ASP programs are based on the stable model semantics of logic programming, which provides a well-understood and mathematically sound basis for knowledge representation and reasoning. Third, ASP can be used to solve problems in a wide range of AI applications, including planning, scheduling, resource allocation, diagnosis, and prediction. Fourth, ASP is highly scalable and can be used to solve problems with large numbers of variables and constraints.

Also, this work suggests that given enough training samples, a RL driven system that uses a similar architecture will perform better than an adaptive Learning System. Another contribution is the utilization of the user's emotional state as a reward function in an RL-driven ATS.

This work can also be used as a stepping stone to new research. First, by creating courses for different subjects we can evaluate and improve the representational capacity of the current Domain Model. Also, different RL algorithms and methodologies can be used in order to yield better learning gains. Finally, what we achieved with AffLog RL in this work is not personalization, but finding the optimal course for all the users. To achieve personalization we should have a specific policy for each user. This policy could be created by monitoring the user through multiple courses and taking into account the user’s prior experience, as well as the experience of similar users.

The remainder of this dissertation is structured as follows: In the second chapter, an extended theoretical background of the various technologies and methodologies that were used is presented. In the third chapter, we present related and state-of-the-art work. In the fourth chapter, the first system, AffLog is presented and discussed. In the fifth section, Afflog RL is presented. In the sixth chapter we discuss the evaluations of the two systems, as well as the tutoring course used as a use case. The seventh and last chapter concludes this dissertation by discussing the findings in greater detail and, pointing to future work .

Chapter 2
Theoretical Background

In this chapter we present the different technologies and methodologies used in this dissertation as well as their respective fields. First we review Intelligent tutoring Systems, their history, their components, and the main Knowledge representation methods that ITS employ. Second, we describe work done in modeling teaching tactics and strategies by humans for Tutoring Systems. In the third section, we briefly explain what Learning Styles are, and how they are used in Adaptive Learning Systems. In the fourth section we review the field of Affective Computing, including emotion detection, emotions associated with learning, and Affective Tutoring Systems. In the fifth section we briefly describe the Answer Set programming paradigm, and the Event Calculus Action Language. In the sixth section we describe Reinforcement Learning, and the algorithms used in this work.

2.1 ITS Systems

An Intelligent Tutoring System (ITS) is a program that is designed to emulate a human tutor and, furthermore, using Artificial Intelligence to adapt instructions and teaching according to the background of each individual learner. These systems are a combination of the disciplines of Computer Science, Cognitive Psychology, Human-Robot Interaction and Educational research. Some advantages of ITS are that they are location independent, easily accessible, offer high flexibility allowing students to learn at their own rates and not having to rely on rigid classroom schedules.

ITSs are important tools for education because the all-in-one approach of most classrooms is not very effective when deep levels of comprehension are needed. Learning via lecturing usually activates factual and rule-based thinking such as memorizing facts or definitions, but it rarely promotes deep thinking such as problem solving or making bridging inferences. When students have difficulty with subjects that need deep thinking such as mathematics they may turn to one-on-one tutoring such as private classes, which is far more effective [10][11]. However, the cost of personal tutors is great and not all students can afford it, leaving them at a disadvantage compared to other students. That's where ITS steps up as a technological solution to this problem.

A typical ITS acts as a personal tutor that employs different teaching strategies in order to teach the user a specific subject. Teaching is usually done by having a virtual human avatar (or agent) present the user with the teaching material, that can be text, diagrams, pictures or video. The avatar is usually capable of speech, and the user can communicate with the tutor through a user interface, input text, or even with his voice. Evaluation of the student is usually done in the form of quizzes. According to the user's answers, the system then adapts to the user, by using different teaching

techniques in order to maximize learning. Affective Tutoring Systems take this adaptation a step further by capturing and taking into account the user's affective gestures while using the system. This can be achieved by devices such as cameras, microphones and biometric sensors measuring the user's blood pressure, heart rate, etc.

2.1.1 Brief History of ITS

The first ITSs were developed and launched in the 70s. The predecessor of the ITS was called computer-assisted instruction (CAI) or computer-based training (CBT). CBT systems started appearing in the early 1960s following techniques such as *Programmed Instruction (PI),* a structure based on a computerized input - output system. This kind of teaching is related to any structured, goal-oriented instruction. As learners were led through the material of a lesson, responses were obtained at every step with a multiple choice test. Incorrect responses were immediately corrected, and learners were informed of their solution accuracy. If their performance was above a preconditioned threshold, learners were allowed to move on to some other content of the lesson. If not, they had to re-study the same material. Most supporters of the PI technology strongly believed that it would enhance learning, particularly for low aptitude individuals however, there was limited evidence supporting its effectiveness. In general, PI refers to any instructional methodology that utilizes a systematic approach to problem decomposition and teaching [12,13].

In the 1970's most CAI systems and ITSs developed further, making use of technologies and techniques such as Knowledge Representation, Student Modeling, Socratic questioning, Bug Library, Expert Systems, and genetic graphs. "Bug Library" [15], is a collection of mistakes in a genetic graph [16], where "genetic" means that knowledge is evolutionary, and the graph denotes the relationships between parts of knowledge expressed as links in a network. Socratic Questioning [14] is an approach in which teaching and learning is performed in the form of question and answer. It is a kind of series of questions in which an original question is split into many low level questions. With this strategy we start from the question which student or learner knows and move to our target questions which we want to teach him.Questioning and answering is structured systematically to reach an ultimate goal.

Cognitive models [17] became widely known in the 80s with other areas of research and development gaining ground such as Natural Language Processing (NLP) [18], authoring shells, fault discovery and predictive modeling [19]. Authoring shells are e-learning systems that feature authoring environments for system creators making software development simpler. Domain knowledge in such systems can be represented by using different knowledge representation specifications [20]. The 1990s saw a lot of multimedia and learner control coming into the system. Moreover, there was a shift towards collaborative learning as opposed to individualized approaches adopted thus far.

In recent years, progress has been made towards providing adaptivity and personalization in computer based education through student modeling, mobile technologies, educational games, serious games and standalone educational applications. The main idea is that an ITS should adapt to the specific needs, knowledge and background of each individual student as they change over time. The major approaches introduced were overlay, perturbation, stereotypes, machine learning techniques, cognitive theories, constraint based models, fuzzy student modeling, Bayesian networks, and ontology student modeling.

ITSs that have been successfully implemented and tested have produced learning gains with an average effect size of one sigma, which is roughly equivalent to one letter grade (10%) when compared to classroom instruction and other naturalistic controls [21]. The ITS effect size is greater than the 0.39 effect for computer-based training, 0.50 for multimedia, and 0.40 effect obtained by

novice human tutors [21][22]. It is, however, less than the 2 sigma effect obtained by expert tutors for mathematics in naturalistic contexts [23]. The naturalistic setting is important because ITSs and accomplished tutors have produced equivalent learning gains when face-to-face communication is replaced with computer-mediated communication [22]. Generally, it seems that ITSs are highly effective in helping students learn.

2.1.2 Components of an ITS

A typical ITS consists of the following essential components, [24] : The Domain model, the Student model, the Tutoring/Explanation model, and the User interface model.

The Domain model (a.k.a the expert knowledge model) represents the subject to be taught. It consists of the concepts, rules, facts, problem-solving strategies, the structure of the topics of the domain and its heuristics. This component analyzes how well a student has understood a particular domain/concept. It serves as the main expert knowledge base as well as a standard for evaluation of the student's performance and diagnosis of errors.

The Student model extends the domain model emphasizing on cognitive and affective states of the student in relation to their evolution as the learning process advances. It also acts as a knowledge base containing what the student knows. As the student works step-by-step through his problem solving process, the system starts to trace and record that process. Anytime there is any deviation from the predefined model, the system reports an error.

The Tutoring/Explanation model, (also called *teaching strategy or pedagogic module*) makes use of information from the domain and student models and devises tutoring strategies and actions. This model regulates instructions to the student by utilizing information about his and its own tutorial goal structure in order to devise the teaching activity to be presented. It tracks the learner's progress, builds a profile of strengths and weaknesses relative to the production rules (termed 'knowledge-tracing'). At any point in the problem-solving process the learner may request guidance on what to do next, relative to his current location in the model. In addition, the system recognizes when the learner has deviated from the production rules of the model and provides timely feedback for the learner, resulting in a shorter period of time to reach proficiency with the targeted skills. The tutor model contains several alternatives to put a concept into two states Learned and Unlearned. If a student successfully solves a portion the system updates the probability of reaching the learned state and makes it higher. Systems continue this process until the students achieve a satisfactory state [25].

The User interface model is the interacting front-end of the ITS. It utilizes all types of information needed to interact with learners, through graphics, text, microphones, cameras, biometric sensors and other input devices. Many ITSs use a digital avatar that acts as the instructor, often being able to talk and make facial gestures.

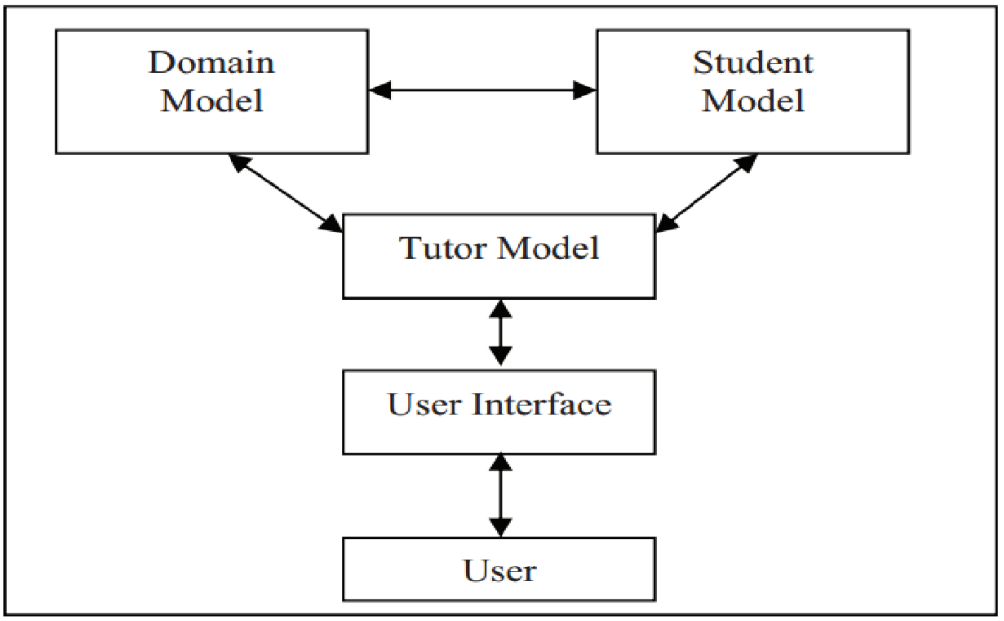

Figure 2.1: Typical architecture of an ITS system [26].

2.1.3 Knowledge Representation in ITS

The domain model is common to all the users of the system, while the student model differs for each user. The student model and the domain model are used together to give feedback to the users. There are three approaches used to build the domain model: Rule Based Models, Constraint Based Models and Dialogue Based Models.

2.1.3.1 Rule Based Models

Rule Based Models, also known as Cognitive tutors, the basis of rule based (RB) models is the Adaptive Control of Thought-Rational (ACT-R) theory of cognition [27]. According to this theory, there are two types of knowledge that a student can learn: declarative and procedural. This theory says that human learning goes through various phases. The first phase involves learning *declarative knowledge*, which represents the overall knowledge of a particular domain. This declarative knowledge is then utilized by the user to solve problems within a domain. This problem specific knowledge is called *procedural knowledge*. The procedural knowledge is goal-oriented. Such goal oriented knowledge can be represented as a set of rules.

The Rule Based (RB) system is based on a set of rules where a user follows a step by step problem solving process. The system then provides a response to each step that the user takes to solve the problem. Because of this, the cognitive tutors are said to generate immediate feedback. When a step does not match any of the valid rules, an error is detected. The rule-based models are built from cognitive task analysis, which is the production of task models by observing the expert and novice users. Task models represent a set of production rules in which each rule represents an action corresponding to a task [26]. When a user tries to solve a given task, the user's solution is compared to the solution given by the expert. This process is termed Model Tracing. During teaching, the Tutor suggests to the user the next step that should be taken. It also provides demonstrations, and it evaluates the understanding of the user in terms of the skills that the user has applied. Finally the system is able to Infer the goals of the user.

These kinds of tutors are basically used if the learning process is of utmost importance rather than simply checking whether the user has obtained the correct answer or not. There are many tutors built using this model. For example, the Andes Tutor for Physics [28]. A problem with these tutors is that the domains have to be well defined so that clear strategies will be available. Moreover, for complex domains, a large number of rules have to be used to represent the knowledge making rule creation a long and difficult process.

2.1.3.2 Constraint Based Models

While Rule-based models capture the knowledge, which is required to generate the step-by-step solutions, Constraint-Based Models (CBM) express the requirements which all solutions should satisfy. This means that the rule based model analyzes the path through which the solution is obtained while constraint based models analyze only the obtained result. Ohlsson's theory of learning from performance errors [29] gave rise to this method. This theory suggests that even when the users have been taught the correct way of performing a task, they still make mistakes. The reason for making mistakes is that the declarative knowledge that the user has learned has not been converted to procedural knowledge.

Practice and catching mistakes can help the user in modifying the procedure. This helps the user to incorporate the appropriate rule that has been violated. The process of learning from errors consists of two phases as described by Ohlsson. These two phases are error recognition and error correction. Error recognition means identifying an error. The ITS recognizes an error by using declarative knowledge. After recognizing an error, it must be corrected. An ITS can perform the role of the mentor to inform the user of the various mistakes, if he/she does not possess the declarative knowledge to identify it. A student often requires the help of a teacher to overcome a problem in his/her own understanding about the particular concept. This can be achieved in CBM by a series of carefully designed sequence of feedback messages. Thus in this way, the CBM reflects the action of a human teacher who helps the student to overcome problems in his/her knowledge. In CBM, a solution is specified by a set of constraints instead of providing an explicit task model. Each constraint consists of a relevance condition and a satisfaction condition. SQL-Tutor is an example of an ITS that uses CBM wherein the domain model consists of over 500 constraints [30]. When the user violates any constraint during solving a task, the CBM Tutor diagnoses that an error has been made and provides help to the user regarding the violated constraint [31]. When a user violates a constraint, it simply means that the user does not know the concept and needs help.

2.1.3.3 Dialogue Based Models

In Dialogue based tutors, systems hold conversational dialogue with its users. There is an agent that acts like a teacher and converses with the user. In addition to displaying graphics, text and animation, this agent also has synthesized speech, facial expressions and gestures. Dialogue based tutoring systems present challenging problems and questions to the user. The user then types answers in English, and there is a lengthy multi turn dialogue between the user and the agent until complete solutions or answers evolve. Users type their responses in English in addition to the conventional point and click. These tutors help the users to actively develop their knowledge through conversation. Some of the successful dialogue based ITSs include AutoTutor [31], ITSPOKE [32], Tactical Language and Culture System [33] and others. These different tutors vary in the extent to which they simulate human dialogue mechanisms, but all of them attempt to comprehend natural language, formulate adaptive responses, and implement pedagogical strategies to help students learn.

2.2 Modeling Human Teaching Tactics and Strategies for Tutoring Systems

Despite the successes of the existing ITS systems, the teaching competence of those systems has been questioned by the pedagogical community, and the question of whether these systems could and should mimic human teachers has surfaced. Boulay and Luckin,[34] have explored certain aspects of human expert teaching and how it was implemented in ITS. Also, they investigated how those systems deal with student answers as well as with motivational issues. Finally, they highlighted the plausibility Problem [35] which is concerned with whether tactics that are effectively applied by human teachers can be as effective when employed by machine teachers.

The main critique of the early Tutoring systems was that teaching involves a wide variety of communicative activities such as explaining, persuading, arguing, demonstrating, describing and so on, and that a teaching system would first implement these general skills and then specialize as needed for the particular educational context at the time. However, the theories of teaching that were implemented in tutoring systems were not grounded in such general communicative competence (because it was beyond the state of the art), so they necessarily treated “teaching” as an isolated and largely self-contained skill.

Examples of that is Socratic Questioning, and various systems produced by Anderson and his colleagues [36][37] which monitored the student’s problem-solving actions and had the capability of reacting immediately if the student departed from the path that an ideal student would have followed, which is something that an expert teacher would not do. So, despite ITS developing effective strategic and tactical means-ends rules for the use of the teaching actions, they have not concentrated on the following:

- The development of a variety of teaching actions.
- The development of basic communicative and competence skills such as explaining, arguing, convincing, dealing with misunderstandings and interruptions, etc.
- The development of theories of motivation and affect that would enable an effective change of topic, a use of a joke, an imposition of a threat, an offer of praise and so on.

There are three principled methodologies for developing the teaching expertise in ITS systems.

- the observation of human teachers followed by an encoding of effective examples of these teachers' expertise, usualy in the form of rules. Two important aspects of this methodology include how to motivate students and how to deal with student errors.
- A methodology derived from learning theories for creating a teaching theory.
- The observations of real students or simulated models of students used to derive a teaching theory from experiments with such students or models of students.

2.2.1 Observation of human experts

While the field of education has studied the skill of teaching for centuries, much of its work is at a level that is hard to implement in ITS, with Socratic Tutoring as an interesting counter-example. But there was an increasing body of work that had observed and codified expert teaching. From these records it is possible to extract general teaching strategies and specific teaching tactics, as well as to compare and contrast these with those available in existing systems.

VanLehn (2011) [38] provides a meta-review comparing the differences and effectiveness of mostly

non-expert teachers and ITS. His paper states the possible reasons why human tutoring might in principle be more effective than computer tutors. These reasons include: detailed diagnostic assessments, individualized task selection, sophisticated teaching strategies, learner control of dialogues, broader domain knowledge, motivation, feedback, and the potential for the tutor to elicit effective learning behavior in their students.

2.2.2 Tactics taken from learning theory

First there are the epistemological theories where the focus is on the subtle way that information is transformed into knowledge and then knowledge into understanding and skill. For example, Contingent Teaching, [39], acknowledges the learner's need for independence and tries to maintain the learner's attention in a learning interaction by providing only sufficient assistance at any point to enable the learner to make progress on the task. The evaluation of this strategy in the hands of non-teachers who had been deliberately using it shows that it is effective.

From the ACT and ACT* theories, [40], came the idea of the transformation of declarative knowledge into procedural skill and thus the value of setting goals, graded exposure to more complex aspects of the domain by using model-tracing and knowledge tracing, and immediate feedback on errors so that they could be acted upon soon after an error had been committed.

Second, there are the reflective theories that in different ways present the idea of two complementary psychological processes operating within a learner: the one focusing on the domain of the subject and the other reflecting on how far that focus on the domain is leading to secure understanding [34]. From this insight various ways for a teacher to support each process were derived, but particularly the second, the metacognitive one. These theories included Pask's Conversation Theory, [41], Reciprocal Teaching, [42], Self-Explanation, [43], and Self-regulation Winne, [44]. Pask's Conversation Theory described the interaction of the two processes in formal terms. His work was applied in the design and development of various computer-based learning systems some of whose interactions had a reflective component similar to reciprocal teaching, where the learner and the teacher take turns to explain to each other the subject matter to be understood and learned.

2.2.3 Observation of students

Research on student observation is based on individual differences, such as gender, ability and their consequences for differentiated teaching, typically via macro adaptation to the style of interaction, [45]. Macro Adaptation is where a teaching or learning system is adapted, or adapts itself, to one particular group of learners as opposed to another group, for example novices versus semi-experts. This is contrasted with micro-adaptation where the adjustment is on an individual and typically dynamic basis. Recently, there have been many datasets describing the progress of students learning in various settings.

2.3. Learning Styles and Adaptive Learning Systems

In order to adapt to the learning preferences of the user, Adaptive Systems typically make use of a Learning Style to group together similar students.

2.3.1 Learning Styles and the Felder and Silverman model

Learning Style is a representation of how individuals perceive and process information while learning. Researchers have developed a number of learning style theories, [114]. One of them, the Felder and Silverman Model, [1], has been recognised as a highly suitable model for e-learning systems, [115]. The model consists of four "dimensions" of learning styles. The Active/Reflective dimension determines how a student likes to process information. The Sensing/Intuitive dimension determines how a student likes to perceive information. The Visual/Verbal dimension measures the type of presentation material that a student prefers. The Sequential/Global dimension deals with how students prefer to organize and work towards understanding information. Although the Felder-Silverman model is still widely adopted, [116, 117,118], recent works, [119, 120] view Learning Styles as a myth, mainly due to lack of scientific data supporting their evidence. Other prominent Learning Style theories of note include the VARK model, [121], and Honey and Mumford's model, [122].

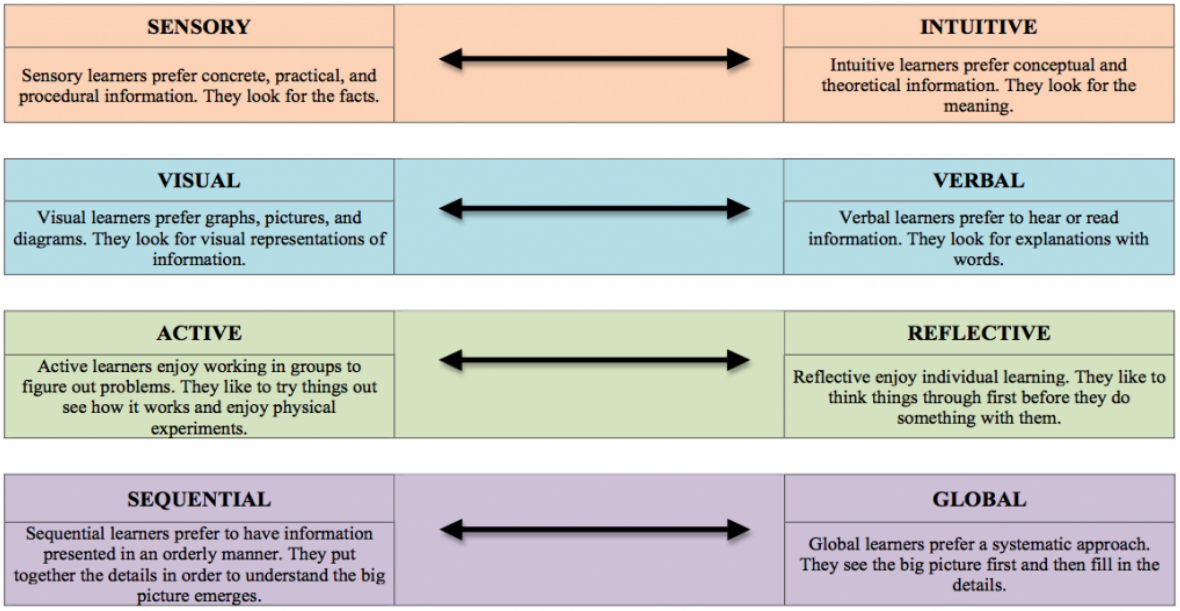

Figure 2.2: Felder-Silverman model dimensions.

2.3.2 Adaptive Learning systems

Adaptive Learning systems, [76] is a category of teaching systems that make use of Machine Learning techniques (including fuzzy decision trees, Naive Bayes, Bayesian Networks, and Neural networks), in order to predict student behavior. Prediction is done by classifying students in certain groups typically according to a learning style theory. Typically, after predicting the learning style of the user, the system will alter the presentation of the teaching material in order to better suit the needs of the user. The most widely used learning theory is Felder–Silverman's model [1]. From this theory, 8 main learning preferences have been designated. A major disadvantage is that such systems do not offer any type of feedback to the user (emotional or cognitive).

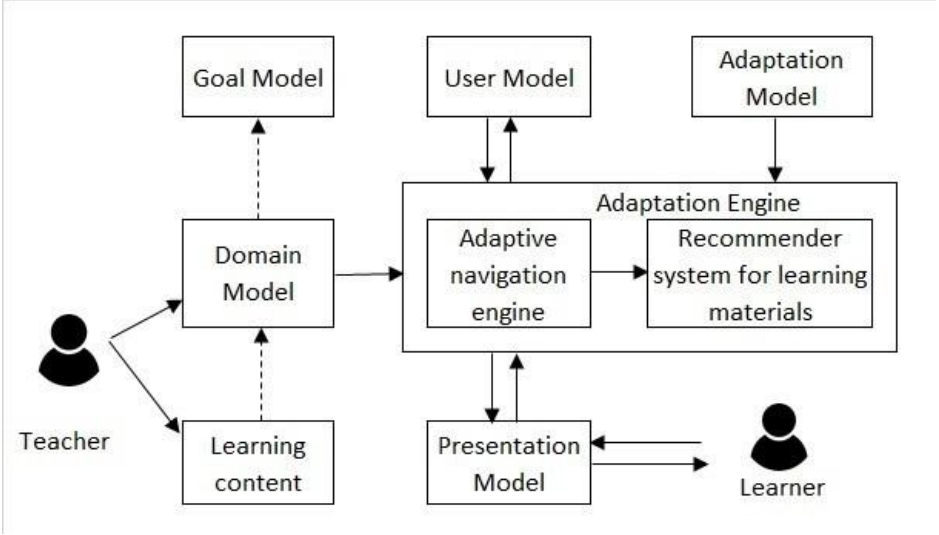

Figure 2.3: Architecture of an adaptive Learning System, [123].

2.4 Emotional Intelligence

Emotional intelligence as an emerging field that deals with modeling, recognition and control of human emotions. Researchers have given a number of definitions of what exactly emotional intelligence is [46]. According to these, emotional Intelligence is defined as: "A set of abilities that involves perceiving and reasoning abstractly with information that emerges from feelings", [47]. And, "the ability to perceive, appraise, and express emotion, the ability to access and/or generate feelings when they facilitate thought, the ability to understand emotion and emotional knowledge, and the ability to regulate emotions to promote emotional and intellectual growth", [48].

Five essential domains for emotional intelligence were defined in [48]: knowing one's emotions, managing emotions, motivating oneself, recognizing emotions in others, and handling relationships. According to the authors, emotional intelligence deals specifically with one's ability to perceive, understand, manage, and express emotion within oneself and in dealing with others.

The First breakthrough of emotional Intelligence was in 1997 when Picard wrote the book "Affective Computing" [49]. In the book, the author presented a framework for building machines with emotional intelligence. In the next few years, many other researchers in this area have developed systems that can reason about emotions, and also detect, handle, understand and express emotions.

Influence of Emotional Intelligence on Intelligent Systems in contemporary research is unavoidable. Modern cognitive psychology considers human emotions to be caused by specific situations. Emotional change can trigger a series of physiological responses through the nervous system, and form a unique subjective experience. This, in turn, may cause external expression changes, in the form of gestures, actions, and so on. Efforts in building emotionally intelligent Systems have been concentrated on three key efforts: Emotion Detection, Embodiment in a virtual or physical way, and the ability to synthesize emotions.

In the last years, research in artificial intelligence and computer science has addressed modeling and communication of expressive, emotional content (research on affective computing at MIT, Kansei information Processing in Japan). Such research has led to the development of prototype systems

for many different uses: expressive personal assistants, embodied conversational agents (ECAs), virtual environments conveying emotional information for enhanced user experience, robots displaying emotional behavior, Affective Tutors and virtual agents for entertainment (video games, interactive storytelling).

In this perspective, a technological challenge is to build machines capable of reasoning about emotions, predicting and understanding human emotions, and processing emotions while reasoning and interacting with a human user. This challenge is connected to the development and usage of a large variety of interaction systems, including virtual agents, tutoring agents, and personal scheduler agents. With the aim of creating a new generation of emotional interaction systems, the study of affective phenomena has become a "hot" topic in computer science and artificial intelligence.

Different logical methods have been recently exploited in order to provide a rigorous specification of how emotions should be implemented in an artificial agent and how agents should reason about and display some kind of emotions. Although the application of logical methods to the formal specification of emotions has been quite successful, there is still much work to be done. For example, there exists no formal model capable of adequately characterizing complex emotions such as regret, jealousy, envy, shame, guilt, reproach, admiration, remorse, pride, and embarrassment. These emotions involve very sophisticated forms of reasoning, such as self-attribution of responsibility, counterfactual reasoning, reasoning about norms and ideals.

Recently, a lot of evidence has been gathered to suggest that virtual agents induce positive feelings in humans during interaction if the agents are capable of displaying emotions, as well as recognizing and responding to human's emotions. In turn, this also improves the virtual agent's performance. Much research has also been invested into building emotions into agents, resulting in a number of formal models for emotional intelligent agents, [59, 60]. Finally, some challenges of developing successful embodied conversational agents are outlined in [61], namely the lack of a solid psychological foundation.

Interesting research questions of that field include exploiting logical methods in order to provide a specification of how emotions should be implemented in an artificial agent. The design of agent based systems, in which agents are capable of reasoning about and displaying different emotions, can benefit from the accuracy of logical methods. Another research question is identifying desirable features of emotion theories that make them ideal blueprints for agent models. These new findings will be introduced into the design, in order to explore their application in concrete areas such as services of tourism, education, or in recognition of infants' emotions.

2.4.1 Emotion detection and input modalities

Modern cognitive psychology considers human emotions to be caused by specific situations. can trigger a series of physiological responses through the nervous system and form a unique subjective experience. This, in turn, may cause external expression changes, in the form of gestures, actions, and so on. By detecting different physiological signals and external features, it is possible to compute and identify the emotional state of a subject to a certain extent. The main interaction methods that machines can use to detect human affective displays include *speech*, *human body language* and *facial expressions.*

An emotional Intelligence System needs to be able to understand *facial expressions*, *body language*, *voice*, and *physiological signals* in order to be able to map them to a certain affect or emotion [51]. There are three major types of models that are used to describe affect: Categorical, dimensional and componential. Whereas Categorical models consist of discrete affective states (happy, sad, Angry, etc), dimensional models take continuous values and use feature vectors to represent

multidimensional spaces that allow the representation of varying intensities of affect [52]. Componential models are based on the appraisal theory [53], and are created by constantly evaluating a subject's internal state and the state of the outside world. As of today, there has not been any automatic system utilizing this theory due to its complexity.

The most widely used dimensional model is Russel's Circumplex of Affect, [54]: According to it, each basic emotion can be represented by two vectors. The first one indicates *arousal* (aroused vs. relaxed), and the second one *valence* (pleasant or unpleasant). Other dimensional models introduce a third vector (dominance-submissiveness), [55], and some make use of an expectation vector, [56], and intensity, [57]. In general, most studies make use of categorical models. Also, most systems are able to understand Ekman's six emotions (joy, surprise, sadness, fear, anger, disgust), [58].

Figure 2.4: Ekman's six emotions (joy, surprise, sadness, fear, anger, disgust), [58].

All affect recognition systems focus on a more specific domain such as facial affect recognition, body language-based affect recognition, Voice-based affect recognition, affective physiological signals recognition (such as heart rate, tension in specific muscles, breathing rate, etc), and multimodal affect recognition that combines one or more of the above. Nearly all systems use Classification Models in order to predict a certain affect. Also, many systems make use of databases containing relevant information, [51], either for representing ground truth or to test their results .

Most systems that perform *f*acial affect recognition use 2d cameras to detect facial features and then classify them using Machine Learning algorithms such as binary decision trees, AdaBoost, Neural Networks, SVMs, Support Vector Regression or dynamic bayesian networks, [51], with the SVMs being the most popular method. The greatest challenges of this approach are the lighting conditions and working distances of the cameras, the real-time computational requirements, and how to cope with spontaneous affective expressions. Affectiva's Affdex api, [96], is a very popular tool for facial affect recognition ,and many demos and tutorials can be found online.

Body language contains information about a person's affect. More specifically, body postures and movements, such as torso orientation and arm positions can display affect during social interactions, [97]. Although most works in HRI were concerned with recognizing hand postures and gestures as input for commanding a robot, a few have been done specifically to identify body language of individuals, [51]. Again, most methods try to extract different features and then apply Machine Learning Classification techniques in order to identify specific gestures. The most popular approaches use SVMs or KNN algorithms. Gestures can be perceived as happy, angry, sad, polite,

and as gestures of approval. Most methods following this approach use the Kinect sensor, [98], to get a 3d representation of a body.

Voice Based Affect Recognition is also feasible since features of vocal intonation that directly correspond to emotions can be extracted by sound processing methods, [99]. Most popular features include: The mean and range of the fundamental frequency, the intensity (total energy), utterance contour, high frequency energy, and word articulation rate, spectrum shape, speech rate, and intensity of a voice signal, [100]. Popular classifiers for these tasks are SVNs, and GMMs (gaussian mixture models), [101]. A dominant force in this field is the Open EAR Open-Source Emotion and Affect Recognition Toolkit, [102].

Studies have shown that monitoring a person's physiological signals such as his heart rate, skin conductance, ElectroDermal Response (EDR), tension in specific muscles, breathing rate, and others, [97, 103], can produce information about that person's feelings. The main problem of these approaches is that most of them are invasive or require the person to wear special equipment.

Finally, researchers have used multimodal techniques, utilizing more than one of the methods described above in order to get more robust and complementary information. Although those techniques outperform all the others, they require more than one sensor and need to be far more complex in order to fuse and process data from different sources, [104].

Although there are many databases representing each modality separately [51], [101], only few deal with multimodal affect data. Most of them contain audio-visual data of various conversations annotated in order to provide continuous values for metrics such as the various affectual dimensions. These include the (SAL) Database [105], The Vera am Mittag Database [106], the SEMAINE corpus [57], and more recently, the HEU emotion video database, [107].

2.4.2 Emotions and Learning, ATS.

Evidence exists supporting that emotion can influence aspects of cognitive performance and decision making, [91]. More specifically, affect can provide information such as context and meaning to situations as well as indicate a subject's attention, [92]. Also, negative emotions are known to interfere with the ability to process new information, [93], and they can often be indicators that something went wrong with the processing of information, implying that future processing should be more systematic, rational, and slower, [94]. So, it is evident that the emotional state of a student can directly influence his learning experience.
Research has shown that a learning session will be improved if the teacher is empathic to the emotions of the learner. This applies to both human-human and human-machine interactions, [95], [31]. Such improvements could be the offering of help when the tutor detects confusion, or giving motivating comments when it detects boredom.

Intelligent systems that make use of emotional information are known as Affective Tutoring Systems (ATS). These systems combine tutoring strategies, affect sensing, and learner progress tracking into a single system. Also their evaluations have shown that they can contribute positively to the learning experience. An ATS has the same architecture as an ITS, with the addition of the affective module, which is responsible for updating the user’s current emotional state in the student model, and also for alerting the Tutoring module whether an affective reaction should be taken in response to the emotional state of the user. An example of affective reactions can be seen in the scenario depicted in figure 2.5, where, the authors of [110] tried to describe a tutor's best course of action given the circumstances.

Cognitive clues	*Affective clues*	*Tutor intervention based on inference about a student's state*
Student makes an error	Student appears curious and focused	*No intervention needed;* Student is engaged in learning and exploration (Flow)
	Student is frowning, fidgeting, and looking around	*Alternate actions are needed;* Student is confused (Stuck)
Student has not made much progress	Evidence of stress, fidgeting, high valence and arousal	*Alternate actions are needed;* Student is under stress (Stuck)
	Evidence of boredom and confusion	*Interventions using off-task activities are needed;* Student is not engaged (Stuck)
	Student is not frustrated	*No intervention needed;* Student is curious and involved in exploration (Flow)
Student is solving problems correctly	Student is not frustrated and is engaged	*No intervention needed;* Student is in control, concentrated and focused (Flow)
	Student is bored – problems are too easy	Escalate the challenge for a bored student

Figure 2.5 : Affective Interventions according to cognitive and affective clues, [111].

Most learning theories in Tutoring Systems deal with the thinking and learning processes of the learner marginalizing or ignoring the learners' affects altogether. An exception to this is the Theory of Flow [50]. Flow theory suggests that when one is actively engaged in an activity where the skills possessed are balanced to the challenge of the activity, he can reach an optimal state of experience called "flow." Several conditions contribute to this psychological state. Also, researchers have found certain emotions that correlate positively or negatively with the learning process. Specifically, Craig et al, [30], in 2004 identified six main affective states during a learning session with the AutoTutor system (frustration, boredom, flow, confusion, eureka, neutral) and found a positive relationship between both confusion and flow and learning, and a negative one between boredom and learning. Eureka represents positive surprise and possibly a breakthrough.

The topic of which emotions are associated with learning has been investigated to a certain degree. In 2001, Kort et al., [108], developed a two dimensional model that linked emotions with stages of learning and proposed approaches to validating it. Plasset al in 2014, and Um et al in 2012, in their respective works, [109-110], showed that positive emotions generated during multimedia learning facilitate comprehension of material and increase the effort, motivation, and overall satisfaction of the learners.

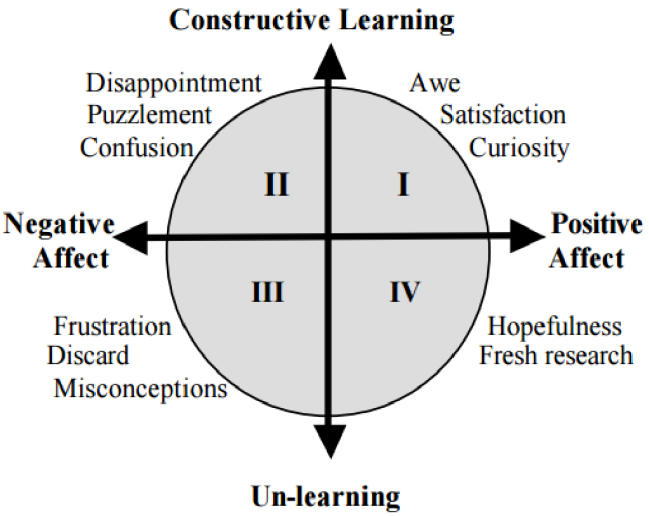

Figure 2.6: Kort et al., [108] Model relating emotions with Learning.

Although there are a number of affective recognition APIs such as those shown in 2.4.1,they do not correspond to learning emotions. Hence, many ATS works have built their own emotion recognition frameworks suited to recognize emotions related to learning, however, these frameworks are closely embedded to their respective ATS and therefore cannot be reused making the creation of new ATS a complicated process.

2. 5 Event Calculus and Answer Set Programming

The Event Calculus [3,6] is a first-order logical formalism for representing and reasoning about actions and their effects similar to the situation calculus, [112] . It is able to form a narrative chain of events using discrete timepoints to represent the effects of actions and fluents. This work implements a translation of Discrete Event Calculus (DEC) theories into ASP rules, which are then executed by the Clingo, [5] ASP solver. Kim et al. translation of DEC was used, taken from the f2lp System.

Answer Set Programming (ASP), [2] is a declarative problem-solving paradigm oriented towards complex combinatorial search problems, based on stable model semantics, [2] . A domain is represented as a set of logical rules, whose models, called answer sets, correspond to solutions to a reasoning task such as progression or planning. Sets of such rules are known as answer set programs. Besides explainability and transferability, ASP offers a number of advantages. Firstly due to its high expressiveness and declarativity it is easy to model with it the various parts of a Tutoring System including constructs such as constraint rules, soft constraints and cardinality constraints. Secondly, one can treat non-deterministic models by utilizing multiple answer sets, where each answer set corresponds to a possible solution.

ASP programs, allow us to express in a mathematical sense, every property of finite structures over a function-free first-order structure that is decidable in nondeterministic polynomial time with an oracle in NP. (i.e., ASP captures the complexity class $\Sigma_2^P = NP^{NP}$). Thus, ASP allows us to encode programs which cannot be translated to SAT in polynomial time. For more information on ASP and rule syntax please consult [5].

2. 6 Reinforcement Learning

Reinforcement learning (RL), [83], is a computational approach to goal-directed learning and decision-making. It is distinguished from other computational approaches by its emphasis on learning by an agent from direct interaction with its environment. RL uses a formal framework to represent the interaction between a learning agent and its environment in terms of states, actions, and rewards. Its main features include a sense of cause and effect, a sense of uncertainty and nondeterminism, and the existence of explicit goals.

RL systems typically consist of a policy, a reward signal, a value function and model of the environment. A policy is the core of a RL agent as it defines the agent's way of behaving at a given time. Specifically, a policy chooses the best possible action given a certain state. Each action receives a corresponding reward signal from the environment, that can be positive or negative depending on the action's immediate result. Typically the action that results in the goal state will receive the max reward. Whereas a policy gives an agent the best immediate action, the value function calculates the total amount of reward that agent can expect to receive, by following a certain policy starting from the current state. In other words, values indicate the long term desirability of states, and can be used in order to find the optimal policy that will solve the RL problem. The model of the environment can be used to better guide the agent by taking into account the ramifications of actions. The RL task can be modeled as a Markov decision process (MDP), [63]. A discrete-time stochastic control process that provides a mathematical framework to the task.

One of the problems RL systems face is that they need many interactions in order to converge to an optimal policy. One approach to solve this problem is Interactive RL, [113]. Originally used in robotics, this approach uses a parent-like advisor to support the learning by providing social feedback, evaluative feedback, advice or instruction. The advisor can be a human, or even another RL trained agent. Another solution is Relational Reinforcement Learning, [7]. To learn tasks as rapidly as possible, we need a highly compact representation of the model, and thus we use relational models. These models use a relational representation format for states and actions, and use this format to improve learning. For example by simplifying action-state updates or, by generalizing over different objects of the same type, thereby reducing the learning complexity of domains.

ITS have used RL in order to improve their teaching strategies, [124, 125, 126]. RL has also been used with Affective Robotic companions, [127] that teach skills. However there is no related work that involves Interactive Relational Reinforcement Learning and Affective Tutoring systems. The closest to this is the work of Nickles and Rettinger, [128]. They have built a framework of learning concept semantics using Interactive, and relational RL, while using ASP and the EC to model the environment. Also, the RL agent they employ engages in dialog with human users in order to facilitate learning.

44

Chapter 3
Related Work

In this chapter a selection of Intelligent Tutoring Systems, Affective Tutoring Systems, Adaptive Learning Systems, and Tutoring system builders is reviewed. In the first chapters, we review a number of systems that influenced the work described in this thesis in chapters 4-6, while, in the last chapter we briefly review a number of systems that were found by the author durring or after the completion of this work.

3.1 GIFT

GIFT, [9,79], is an open-source, modular, framework that provides tools, methods and services to augment training applications for creating intelligent and adaptive tutoring systems. It has an online cloud mode useful for e-learning applications and an offline version. GIFT includes: A standard interface, domain knowledge representation (including an authoring tool), performance assessment, a pedagogical model that includes micro and macro adaptation, learner modeling, survey support, and a standardized approach for integrating physiological sensors to the system. Although not dealing with the emotional part of the users, GIFT is modular and flexible. Moreover, it is free, and there is a thriving community of users utilizing it and extending it. Several works using GIFT were evaluated and have displayed high learning gains, [79].

3.2 AutoTutor and Affective AutoTutor

AutoTutor, [30,31], is an ITS that helps students compose explanations of concepts in physics, computer science, and critical thinking by interacting with them in natural language. This is achieved using adaptive dialogues and pedagogical strategies that are similar to that of human tutors, modeling student's cognitive states and using these models to dynamically tailor interactions. Also, AutoTutor can answer student's questions and correct them if they produce a wrong answer. AutoTutor communicates through an animated conversational agent with speech, facial expressions, and some rudimentary gestures.

AutoTutor’s behavior can be characterized by an *outer* and an *inner* loop, where the outer loop consists of a series of didactic lessons and challenging problems or *main questions* and the inner loop consists of the collaborative interaction between the student and the tutor while answering a question (or solving a problem)

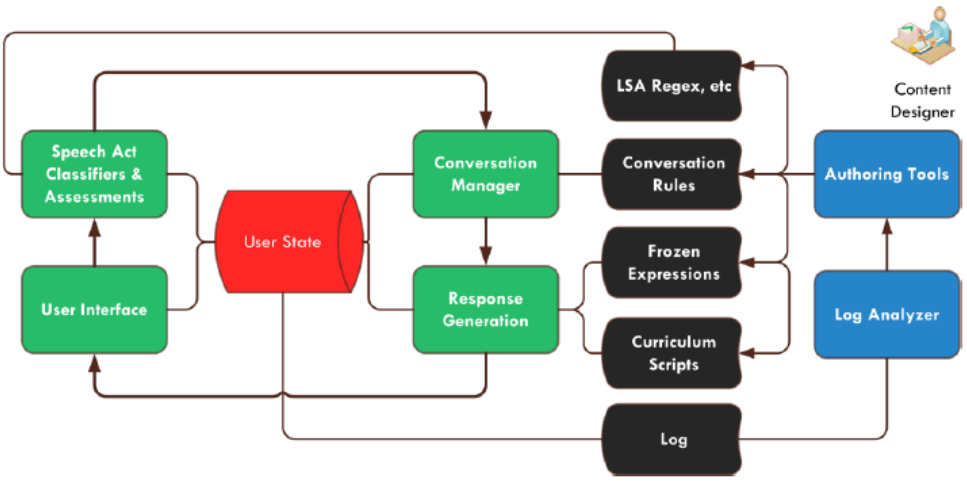

Figure 3.1: AutoTutor Architecture [30].

Student modeling in the inner loop is executed after each turn of the dialog and consists of comparing what the student expresses in language with the list of expectations and misconceptions associated with a main question. This is done using semantic matching algorithms that compare the student input with AutoTutor's expectations and misconceptions.

Autotutor is probably the most famous one of the most widely tested ITS, constantly developed and evaluated since 2008. Learning gains were evaluated to 0.8 sigma. Auto Tutor also led to the development of many variants such as the Affective Auto Tutor.

In order to cope with the emotional part of learning, the authors of Auto tutor created the affective Auto tutor [7], adding an affective loop to the existing architecture. The system mainly deals with the negative emotions of boredom, confusion and frustration. The affective loop consists of the real-time detection of affective states relevant to learning, the selection of appropriate actions to maximize learning influencing the user's affect and the synthesis of emotional expressions to be used by the system's virtual avatar.

Affect detection is performed in a multimodal fashion via machine learning methods applied to the dialogs of the tutoring sessions, body language, and facial feature tracking. To regulate the negative affective states of the user, the system always keeps track of five informational parameters that provide affect sensitivity. These are: The current affective state detected, the confidence level of that affect classification, the previous affective state detected, a global measure of the student's ability, and the conceptual quality of the user's last response.

The last one is obtained by Latent semantic Analysis (LSA) of that response. Using those parameters, the system uses interventions triggered by the values of said parameters. These interventions can be: Feedback for the current answer with an affective facial expression of the avatar, an affective statement coupled with a matching emotional facial and vocal expression of the avatar, or, the next dialogue action to advance the conversation.

The authors have developed two pedagogically distinct variants of Affective AutoTutor. These include a *Supportive* and a *Shakeup* tutor. The difference between these is that the Supportive AutoTutor attributes the students' negative emotions to the material or itself, while the Shakeup AutoTutor directly attributes the emotions to the students. Another difference between the two versions lies in the conversational style. While the Supportive AutoTutor is subdued and formal, the Shakeup tutor is edgier, flaunts social norms, and is witty. For example, a supportive response to

boredom would be “Hang in there a bit longer. Things are about to get interesting.” The shakeup counterpart of this response is “Geez this stuff sucks. I'd be bored too, but I gotta teach what they tell me”.

Many evaluations were performed both on the classifiers, (each separately, and in a multimodal configuration), and on the efficacy of the tutor in affecting learning compared to the normal Auto Tutor. Emotion classification has achieved a 78.8% accuracy (chance=50%), while comparison between the affective and not affective systems showed that whereas the performance of students that already knew the domain of the lesson quite well did not improve, the performance of students that had limited knowledge of the domain was improved.

3.3 Genetics with Jean Tutoring System, [74]

Arguing that there is no widespread use of ATS outside a controlled research setting due to the fact that the components of most of these systems are tightly interconnected and cannot be reused , the authors created an ATS system as proof of concept of an approach intended to equip existing e-learning applications with ATS capabilities improving their learning outcomes. It uses an already tested method the Affective Stack Model, [139], for affective application development enabling a flexible connection between the affective model and the other components of the ATS thus not limiting the system to a particular implementation or a particular instructional Domain.

This ATS teaches the subject of Genetics through a presentation that includes texts, graphics animation based on a Genetics Tutorial found online. The learner's affective state is inferred by an affective platform developed by the authors that uses Skin conductivity and heart rate variability measurements as its modalities, and then uses a dimensional model of emotions to calculate affective *arousal* and *valence*. The sensors used were small electrodes affixed to skin with adhesive tape. This affective system is self-calibrating and signals are considered in the context of the learner’s own recent range of physiological expression rather than comparison against a pre-selected ‘baseline’. Also the system operates in a fully automated real-time mode.

Information of the learner's affective state is used to guide an on-screen animated pedagogical agent Jean. Jean's purpose is to provide guidance and support to the user, and to emulate a human tutor. This is achieved by following a relatively simple instructional strategy:

- If the affective state of the learner is neutral, take no action.
- If Performance is low, offer revision.
- If a negative emotional state is detected (high activation and negative valence) offer Support through empathic feedback to the learner and improve learning and satisfaction. For example: ‘I know the material is quite hard’ or ‘It’s ok if you don’t score full marks, you can always come back to this another time’.
- If a positive emotional state is detected (frustration, or distraction) derived by high activation with high valence), offer motivating remarks .

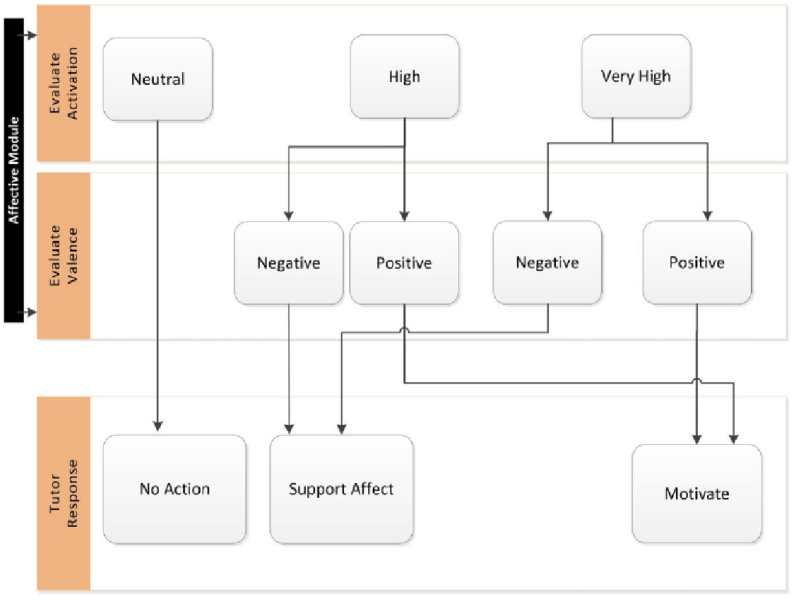

Figure 3.2 : Affective module States and actions, [74].

Evaluation of the system with and without the affective module conducted 40 adult participants. Results showed that using the affective module led to improvement of student learning outcomes. The work done here is important because it demonstrates that a tutorial lacking affect-sensing capabilities can be easily endowed with them, increasing its effectiveness and also because it presents a modular alternative to all the other hardcoded monolithic ATS's. However, the instructional strategy of the pedagogical agent is very simplistic and limited and the sensors used are quite obtrusive.

3.4 Affective Tutoring System for Built Environment Management (ATEN), [140]

The authors of this work have developed this ATS in order to determine the student's interest, stress and learning productivity for a course such as B*uilt Environment Management (ATEN)*, although this approach does not make use of any specific information of that domain, so it could be used for other domains as well. The main innovations of this system are its self-assessment and self-esteem functionalities that can measure a student's feelings and knowledge about a certain topic using questionnaires and monitoring their biometrics with an extended array of biometric methodologies while answering these questionnaires. Those measurements can be used in order to guide the students through their studies.

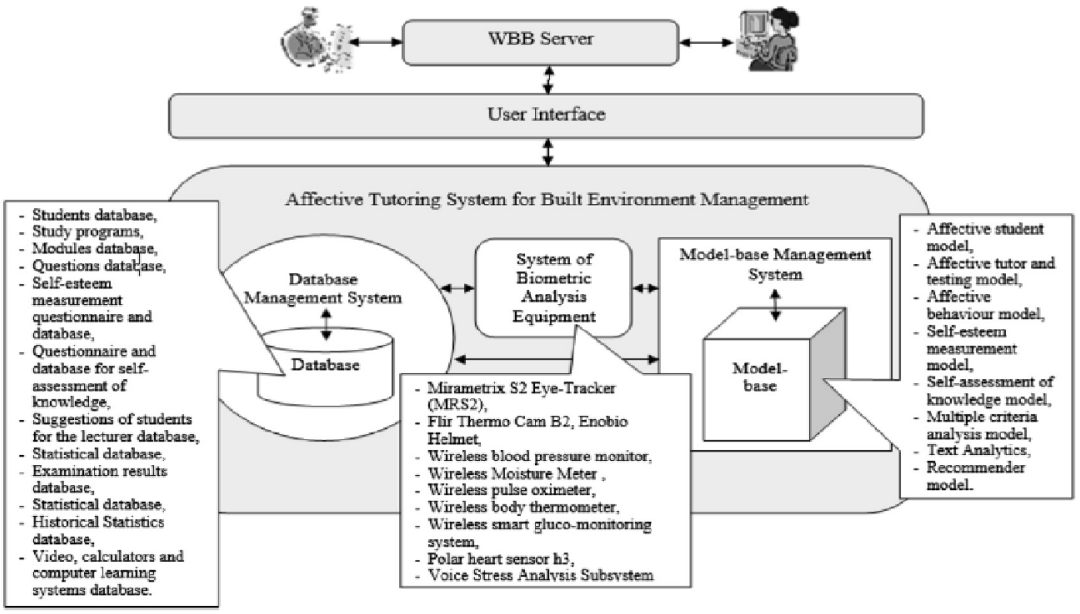

Figure 3.3 : Structure of the ATEN, [140].

The structure of the System, as depicted on figure 3.3 consists of a *Database* containing all the knowledge that the system uses, t*he Subsystem of Biometric Analysis Equipment* that contains all the affect sensing mechanisms, the *Model-Base* and management system and a user interface. The biometrics used are heart rate, systolic and diastolic blood pressure, skin humidity, perspiration, temperature and conductance, Volume Stress Analysis (VSA), EEG. Using various sensors and techniques. The database includes *historical statistical data* gathered from tests such as testing questions, testing results, time distribution for every question and number of times a student changed an answer to each test question, VSA data, correlations between VSA data,correct answers etc. It also lists questions according to modules, possible answers to a question and evaluation of the correctness of possible answer versions.

The Model-base includes the following Models:

The Affective Student Model, that contains information about each student and accumulates information about the whole learning history of the student throughout the semester. The student's knowledge is represented in terms of deviations from an expert's knowledge. This knowledge coupled with the student's levels of interest, stress and learning productivity enable the system to decide what curriculum module, or chapter of a module should be presented to the student next, and how it should be presented (text, multimedia, computer learning system, etc.).

The Affective tutor and testing model is used to decide when to present a new topic and which topic to present. To do this it collects data from the affective student model as an input. After a learning session takes place, the Model compares the student's knowledge before and after the session using quizzes , and then it performs an evaluation based on the differences of the answers. It then gives feedback to a student about his progress and updates the student's affective Model.

The Self-esteem Measurement Model is used to predict a student's behavior, attitude and expectations in an examination. This is done based on previous research of the authors, confirming the hypothesis that students with better grades often underestimate their abilities in self assessments while students with lower grades often overestimate their abilities. Self esteem is measured by giving the student a questionnaire.

The Affective Behaviour Model creates a rational version of the learning process tailored to a specific student, taking into account how much the studies are interesting or difficult and the level

of stress (with the help of biometric technologies). The System includes an automatic function that takes module topics and compiles an optimal set of personalized materials for a specific student. In order to estimate the interest in learning, stress and learning productivity, the model uses a two step procedure. The first step is e-self-assessment, when students assess their own interest in learning, stress and learning productivity, while the system monitors their biometrics. The system then finds the dependencies between the biometrics and the input of the student, until there are statistically sufficient amounts of dependencies. Then, having completed the learning phase, the Affective Behaviour Model later determines the correlation between a student's interest in learning, stress and learning productivity and the biometric parameters of that student. It can then choose specific learning material according to the student's needs.

A workflow of the knowledge assessment process can be seen in figure 3.4. First a student logs in to the system and does the self esteem and self-assessment tests. Then, he must take a multiple questions e-test, and read out loud his answers. The system then gives feedback to the user.

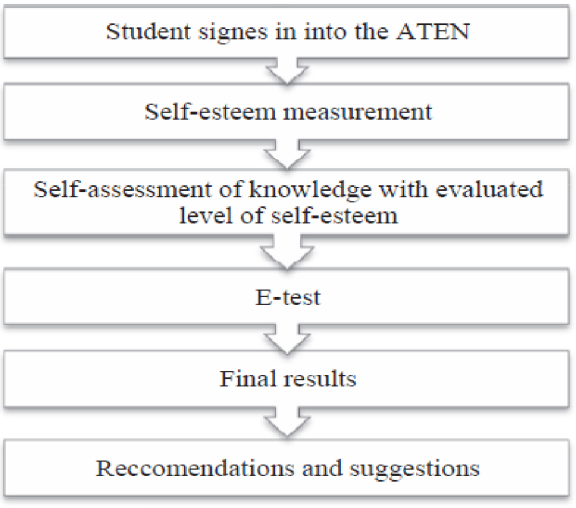

Figure 3.4: Workflow of ATEN, [140].

No evaluation has been performed on how this system improves the learning process of students or helps them with their study. However, a case study with 206 students has shown that there is great correlation between the marks on the e-test and the self-assessment test. This correlation, the writers suggest, could mean that in the future self-assessment tests could replace exams.

Although this work deals with a vast array of biometrics equipment, the tutoring system does not offer any effective help to the students save the selection of literature and if it does, there is no evaluation on its success. Moreover, most of these biometrics systems are quite obtrusive making the adopted use of the system quite difficult. Also, there weren't any examples demonstrating the usage of the system, or how underlying mechanisms such as the selection of appropriate material for each student work

3.5 Other Learning systems.

In this chapter we breifly review a number of interesting selected works, found during or after the completion of this work.

Pandit and Bansala, [129], proposed an adaptive framework for learning using a declarative

approach. This ASP based approach has three main components: The first is grouping pairs of students according to a strategy based on their progress. The second is using planning for adaptive path-finding in order to cover all the topics for learning, which is similar to the work presented in chapter 4 of this thesis. The third component is used to analyze student performance and provide feedback, by taking into account the knowledge gains of the student. Unfortunately the authors did not develop a prototype, so evaluations could not be made.

The authors of, [130], designed an online ATS for curriculum teaching. The affective computing mechanism recorded changes in participant emotions and issued timely reports to an instructor, enabling teachers to provide immediate assistance to participants and understand their learning state. Affect recognition was done either by the users describing how they felt and then the system employed the Semantic Clues Emotion Voting Algorithm, in order to link the description to a specific emotion, or by Facial Expression Recognition using a camera to detect Eckman's six universal emotions. The system was evaluated by comparing it with a non affective version of the system and results indicated high learning gains.

Samy and Naser created the Intelligent Tutoring System Builder (ITSB), [131]. ITSB has two distinct user interfaces. The teacher interface which acts as an authoring tool, and the student interface that acts as an ITS. By the teacher interface, the teacher can add new lessons, adjust the established ones, and revise teaching methods. The student's interface is used to convey all the teaching commands. As evaluation, the authors had a number of teachers prepare a course for introductory Java Programming and a number of students use the resulting ITS. Both groups found the system easy to use, efficient, and friendly. ITSB has been used to create ITS for a variety of subjects such as, among others, introduction to Mondo databases, [133], TOEFL , [134], Cryptography and security, [135], cloud computing, [136], etc.

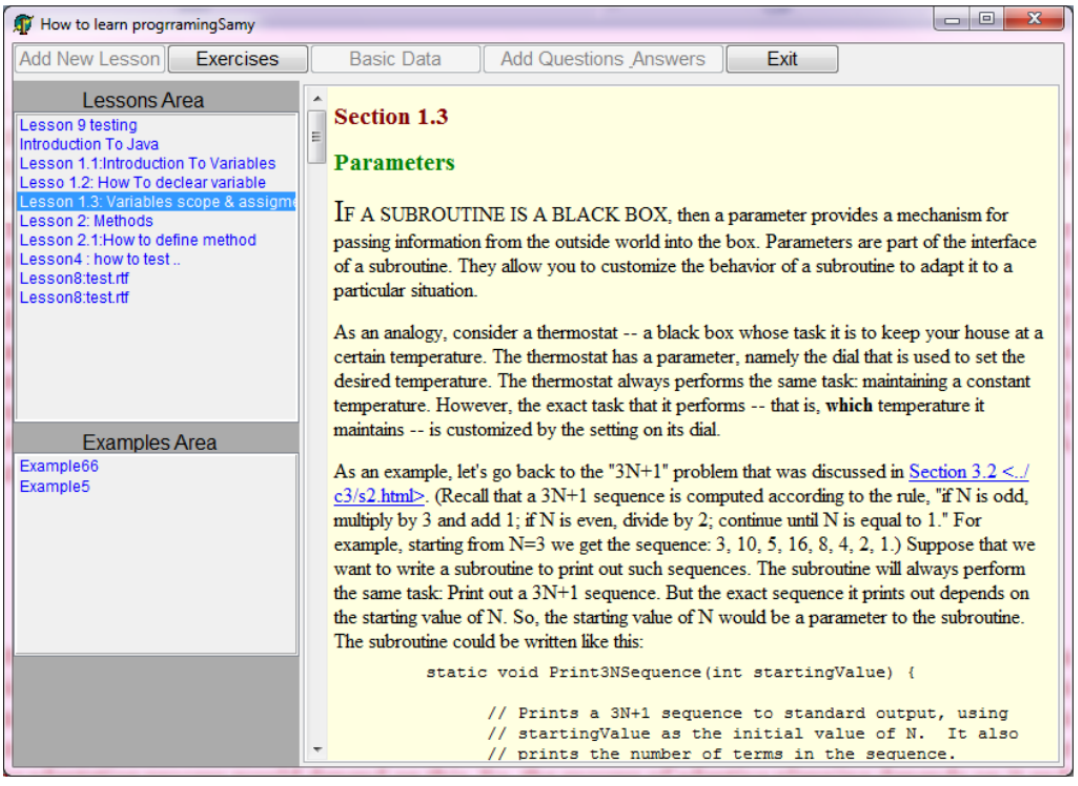

Figure 3.5: ITSB student interface, [131].

The Cognitive Tutor Authoring Tool allows a teacher to include learning by doing online lessons, [141]. It facilitates the creation of two categories of tutors: example-tracing tutors that can be produced with no programming, but need problem-specific authoring, and cognitive tutors, that need Artificial Intelligence programming to create a cognitive model of student problem solving however, provide tutoring across a variety of problems, [141].

The authors of [137], built and evaluated a dialogue-based Mathematics ITS based on diagnostic teaching. This method identifies key concepts and misconceptions, provokes cognitive conflict by presenting open challenges, and resolves problems through discussion. This system can identify students' error types and misconceptions in real-time by using a block-based matching method. The Instructional strategies used in the system were: Presenting open challenges, problem simplification, and representational teaching. Evaluation was done by two groups of primary school students, half of them interacting with the ITS and the other half with a teacher using the same teaching material. Results showed that the first group outperformed the second.

The authors of, [73] built and evaluated a Tutoring system for teaching Algebra that is able to generate practice problems, problem progressions, and step-by-step explanations, by using answer set programming rules. For evaluation, the authors conducted two user studies, showing that users could solve algebra problems more accurately and efficiently after practicing with the tutor, demonstrating that the generated solutions and explanations were understandable.

The authors of, [138], Logic-Muse, a web-based Intelligent Tutoring System that helps learners develop logic and reasoning skills in multiple situations. A catalog of current logical reasoning errors is built, followed by an explicit representation and encoding of the semantic knowledge behind reasoning as well as reasoning procedural structures and meta-structures. The authors identified six classes of reasoning situations, which gradually increase in difficulty, given the nature of the content. The learner model of the ITS keeps track of all the exercises performed by the learner and represents the state of the learner's knowledge. The learner model is a Bayesian network built from domain knowledge, where relationships between nodes (reasoning skills) are provided by experts.

Logic-Muse provides three levels of learning activity to the learner organized into services: 1) Domain exploration service using the domain ontology 2) General exercises on basic logic concepts 3) Meta-cognitive reasoning (logical meta-structure exploration). There has not yet been any evaluation of the system so far.

Chapter 4
The AffLog System

In this chapter, the design and implementation of the AffLog tutor are presented. First, we offer some information about what the system can do and its main innovations. After that, we state some definitions that are used to describe the system. Then, an overview of the system's functionality and architecture follows. After that, we present a section listing the properties used by the system, as well as a detailed explanation of the Answer Set Programs that make use of these properties. Then, in order to get some insight on using AffLog, we consider some examples of its use. Finally, we list some limitations of AffLog in its current form.

4.1 The Afflog System and its Advantages

The Afflog system is a domain-agnostic ATS that can present teaching material to the user adapted to his/her learning style. In order to achieve a domain-independent generalized course structure and also to ensure that the course presented to the user covers all the necessary topics, we make use of Knowledge Representation methods implemented using Answer-set Programming and the Discrete Event Calculus. These same methods also allow for a more complex affective reaction strategy than other ATS, and also allow for a more flexible course structure than the typical linear structures employed by Adaptive Learning Systems.

4.2 Definitions

AffLog : This term refers to the AffLog tutoring system and all its components as an entity.

The user : The person that uses the system in order to learn. For the purposes of this dissertation, we assume that only a single user interacts with Afflog at any time.

A session is an interaction process between AffLog and the user during a fixed amount of time. Interaction data generated during this time is saved in the user's profile. A session ends when the user finishes the tutoring course, or when AffLog decides to terminate the tutoring course (see below), responding to negative feelings from the user. It is also possible for the user to manually end the session. In this case, AffLog saves all interaction data in the user's profile so that it can be

used when the user wants to start another session. A tutoring course may take the user one or more sessions to complete.

The learning material pool: This is a file directory that contains all the learning material that can be used by Afflog in order to build a *tutoring course* for a specific subject.The pool includes *tutorial* files, *test* files, and the *domain mode*l file.

The *domain model* file is a knowledge base that contains all the properties of a specific subject from which tutoring courses can be created. This includes the concepts of chapters, sections, tutorials and tests as well as their properties. It is written in the Answer Set Programming formal language and executed by the Clingo reasoning system, [5], following the stable models semantics. All AffLog Knowledge Representation tasks such as course selection, course planning and projection make use of this file. The domain model properties are explained in greater detail in section 4.4 .

A tutoring course is a selection from the learning material pool that AffLog uses in teaching one or more sessions. This could be an academic course, or perhaps a guide offering instructions for a specific task. It is essentially a collection of parts called *tutorials* and *tests* that are presented to the user. Each tutoring course consists of a number of *chapters* and each chapter consists of a number of *sections*. Sections are traversed using tutorials and can be described as nodes in a graph where the tutorials act as guided edges (Figure 4.1). Sections should not be confused with sessions. The former is a subchapter of a tutoring course, while the latter describes the interaction process between the user and the system. When a chapter has been taught (i.e., when the tutorials that are part of it have been presented to the user), AffLog presents him/her with a test corresponding to that chapter in order to determine whether the user has correctly understood the chapter's content. In order to create a tutoring course, AffLog selects tutorials and tests from the domain model so that all sections found in the domain model are traversed from the selected tutorials and there is at least one test for each chapter.

A *tutorial* is the main building block of a tutoring course. It can be plain text, a picture, an audio file, a video file, or a combination thereof. Each tutorial is represented in the domain model as a logical predicate. Its semantic properties include the sections where the tutorial begins and ends, its modality, its relative difficulty, duration, and learning style properties. Two tutorials may possibly cover the same sections, but their modality or learning style may vary. When a tutorial has been taught to the user it is added to the *state* (see below) of the session along with a tag that describes the user's comprehension of this tutorial from AffLog's point of view. This tag can have three values: Unknown, correct and wrong. The *unknown* tag indicates that although the tutorial was presented to the user, the system does not know whether the user has actually understood it. The *wrong* tag indicates that the system knows that the user did not comprehend the tutorial sufficiently, and finally, the *correct* tag indicates that the system recognizes that the user has sufficiently understood the tutorial in question. Taught tutorial tags can only change when the user answers a test.

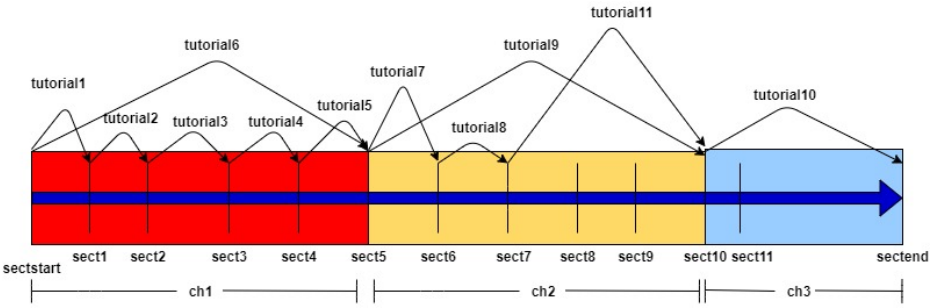

Figure 4.1: A tutoring course as a directed graph. Each section is depicted as a node, with one or more tutorials represented as guided edges. A chapter can be traversed either by one tutorial or by a selection of several tutorials.

A *test* ensures that the user has understood tutorials that belong to the same chapter. A test has a semantic structure similar to a tutorial. If the user passes the test, the system assumes that all the corresponding tutorials are correct, and the system proceeds to the next chapter. If not, then the tutorials corresponding to the test are labeled as wrong. Consequently, the course has to be modified by replacing the wrong tutorials with new tutorials for that chapter. For the purposes of the use case of this work, that is the "Settlers of Catan" course, tests are simple multiple choice questions. When a test is presented to the user, it will appear in the state set (see below) tagged either "right" or "wrong" depending on the answer of the user.

State : By the term *state* we refer to the state of the AffLog session. This is a set of logical predicates that describe the tutorials and tests that have been presented to the user thus far along with their respective tags, as well as the current emotional state of the user.

Action : By the term *action* we refer to what Afflog can do during a session. An action is a logical predicate that can create new state predicates or delete state predicates at a certain time point. Actions include presenting a tutorial to the user, presenting a test to the user, or providing emotional support to the user.

Student Model: The student model depicts the current state of the user. It includes the *state* of the current session as well as information found in a JSON file. The latter contains the user's personal information as well as the user's learning style as described by the Felder-Silverman model [1], (Figure 4.2).

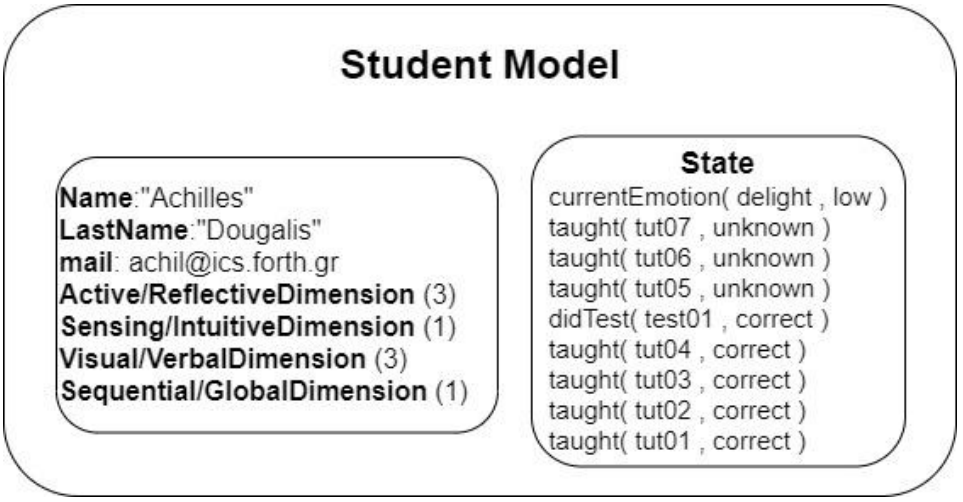

Figure 4.2: An example of a Student Model.

In figure 4.2, the left side contains the user's personal information, while the right side contains the state of the current session. In this particular example, we assume that tutorials **tut01**, **tut02**, **tut03**, **tut04**, and **test01** belong to chapter 1, and tutorials **tut05**, **tut06**, and **tut07** belong to chapter 2. Due to the fact that **test01** is tagged as correct, all taught tutorials that belong to that chapter are also correct. Also, since there are not any other tests done, AffLog does not know if the user has actually understood the taught tutorials of other chapters. Hence, the taught tutorials **tut05**, **tut06**, and **tut07** are tagged as "unknown". The numbers in brackets next to the "dimension" predicates determine the user's score to that particular dimension. For more information about the dimension scores, see section 4.6.

4.3 Functionality

As seen in Figure 4.3, at the first step, the system uses task planning implemented using logic rules written in ASP in order to create all the possible tutoring courses. In this case, each tutoring course created is an answer set derived from those logic rules. The system then selects one tutoring course corresponding to the user's known learning style (we use the Felder-Silverman model [1]) found in the Student model. Specifically, if the user's learning style is closer to global than sequential, AffLog will select long tutorials that contain many sections or even chapters. If the opposite is true, then it will select smaller tutorials. For active users, the system will present more problems to the user in the form of tests, and select the more practical tutorials, while for reflective users, more complex tutorials will be selected. If a user favors visual over verbal learning, then the system will select tutorials with pictures or videos rather than text or audio, etc.

After a course is selected, the tutor starts teaching by presenting the first tutorial to the user. After the user finishes the tutorial he/she records his/her emotional state regarding this tutorial. Afflog then decides whether to react to that emotional state and perform an affective response. The tutor then carries on by presenting the next tutorial until a chapter is finished. At this point, the system needs to find out if the user has comprehended the tutorials taught so far before continuing on the next chapter. In order to do that, it presents the user with a test relevant to that chapter.

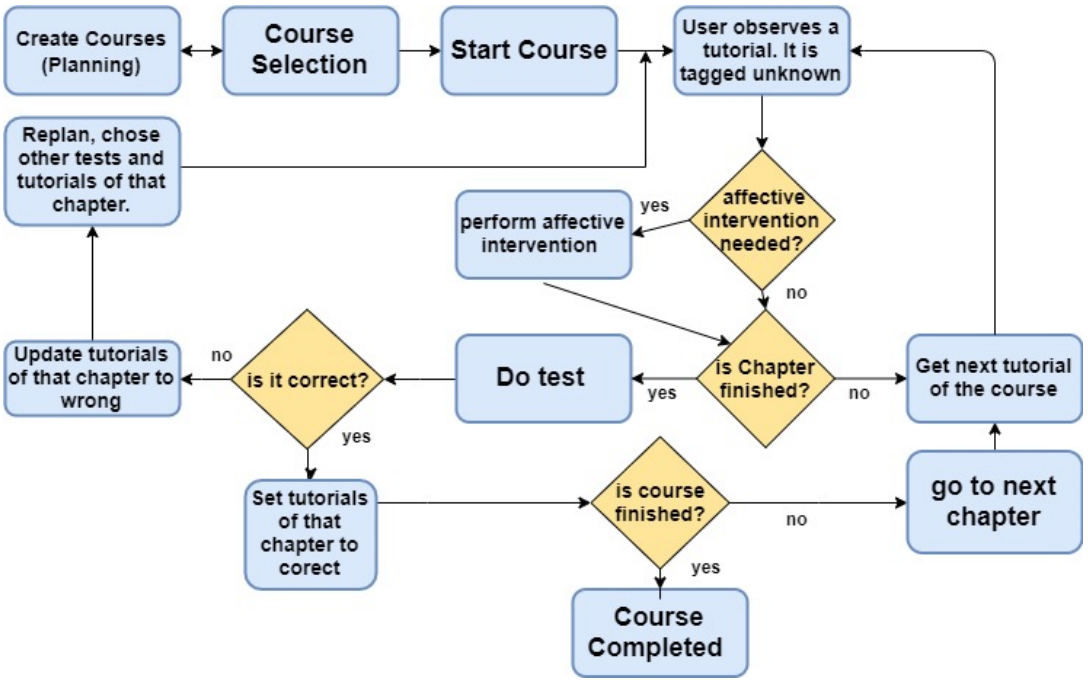

Figure 4.3: Flow chart depicting AFFLOG methods.

If the test has been answered correctly, AffLog will then mark the chapter and its corresponding tutorials as correctly taught in the student model, and the tutor proceeds on teaching the next chapter. However, if the test is wrong, then the tutor assumes that the user did not understand a part of the chapter and explains the correct answer to the user. AffLog then marks corresponding tutorials as wrong and partially plans again the tutoring course, in order to find different tutorials and tests corresponding to that chapter while keeping the already taught correct tutorials and tests. It also tries to select easier tutorials than the previous ones, sharing the same learning style properties if available.

A course finishes when all the primary chapters have been successfully completed and all their tutorials taught correctly. Some courses may contain secondary chapters, in other words, chapters that are not necessary to the task of completing the course, such as examples, exercises, or extra tips and strategies. Such chapters may be added to a course if the user asks for them.

4.4 Affective Response

As seen in section 4.2, during teaching, the affective state of the user may change. In its current form, AffLog does not support emotional detection through sensors. It relies upon the user or an expert to record the user's dominant emotion during a test or tutorial through a user interface. When such a change occurs, the tutor will reason whether an affective reaction needs to be taken and act accordingly. For example, if the user's affective state was neutral but has since changed to confused, the tutor will offer words of encouragement to the user and continue teaching by presenting the next tutorial or test of the course. If the user is still confused during the next tutorial,

the tutor will replan the course from the current state onwards in order to find easier tutorials, thus reducing the user's stress.

AffLog can implement reactions to the emotions of frustration, boredom, flow, confusion, joy and surprise, and will always respond with a light response at the first occurrence of an emotional cue. Light responses are words of praise or encouragement. If this emotion persists for more than a single tutorial or test, then the system will respond with a medium response. Medium responses respond to the user's affective cues by changing the structure of the course accordingly. Finally, if an emotion lasts for three or more consecutive tutorials or tests then the system will respond with a drastic response. Drastic responses will offer to terminate or pause the session giving the user time to recover from whatever problems she/he faces. **Table 1** presents the system's responses according to the user's emotional state. Note that if an emotional cue persists for more than a single tutorial and a medium response is not needed, then the system will again invoke a light response. Likewise, in the absence of a drastic response, the system will offer the corresponding medium response.

Emotional State	Light response	Medium response	Drastic response
neutral	none	none	none
flow	offer praise	suggest extra tutorials	none
confusion	offer encouragement	select easier tutorials	offer short break
joy	offer praise	none	none
surprise	none	select extra tutorials	none
frustration	offer encouragement	select easier or shorter tutorials	terminate session
boredom	offer encouragement	Select harder tutorials	none

Table 4.1 : AffLog's affective responses.

4.5 Architecture

Figure 4.4 depicts AffLog from an architectural point of view. The System's Tutor, affective and UI modules are written in Java, while the Knowledge Representation parts, including the domain model as well as various methods of the tutor module, are written as ASP logic programs using the Clingo System to solve them.

The User Interface (UI) layer consists of Afflog's Interface and a small form that allows the creation of new user profiles. As seen in Figure 4.5, the interface consists of a central section that is used for displaying picture or video tutorials and tests, while the area on the right contains the accompanying text. If a tutorial or test contains audio, then a smaller window with audio controls will appear directly below the central section. On the left of the central section there is a file directory containing all the tutorials and tests of the current tutoring course, while the bar below shows how many tutorials and tests remain in order to finish the course. On the upper part of the interface there is a button associated with creating tutoring courses or replanning existing courses. On its left there is the space where AffLog displays emotional messages. Finally, at the bottom right corner are the buttons used to record the user's emotions, as well as additional buttons used for tests.The user

profile form (Figure 4.6) allows the user to input his/her personal information to the system, including their Learning style, and is stored in a JSON file.

The tutor layer contains all the methods and data structures that are responsible for controlling the system. The tutor layer also contains methods that call the answer set programs through the parsing layer. Among other data, it contains a cache containing all tutoring courses, and structures that contain the current state of AffLog, the student model of the current user, as well as a list of all scheduled actions that Afflog needs to do in order to finish the current tutoring course. It also houses the reactive methods that decide whether Afflog should perform an affective reaction. From a traditional ATS architecture perspective, the tutor layer contains the tutor model, the student model and the affective model of the ATS.

The parsing layer contains all the methods that connect the tutor layer with the Answer Set programs. These methods create and write the input files of the Answer Set programs according to the tutor layer but are also responsible for receiving the output of the Answer Set programs and transforming it to java data types, such as int, String, etc.

Afflog makes use of four Answer Set Programs. Each of these programs consists of a number of files with the .lp extension. Some of these files such as the domainModel.lp and DEC.lp files are used in more than one of these programs.

The “Plan tutoring courses” program, which we will be referring to in this dissertation as the “planner”, is responsible for creating all the possible tutoring courses that are available in the domain model file. These courses reach the tutor layer through the parsing layer and are saved in the tutoring courses cache. The planner uses Discrete Event Calculus (DEC) events and fluents to model AffLog’s actions and the states that are the results of these actions. The planner starts at the zero state, which denotes the absence of all tests and tutorials. It then selects and performs a single action in order to reach the goal state. If the new resulting current state is not the goal state, then it performs another action, until the current state is the goal state. The goal state is the predefined end of the course. This can be the end of the last primary chapter, or the end of the last secondary chapter depending on the preferences of the user. The output of the planner is a set of answers, where each answer is a unique sequence of actions that form a tutoring course. For more information on the planner, see section 4.6.1

The “tutoring course selection” program takes the profile of the user, the domain model and the available tutoring courses found at the tutoring course cache as input in order to select a single tutoring course that better fits the learning style of the user. This selected course becomes the current tutoring course in the tutor layer. This is done by first calculating the scores of each learning style dimension for each tutoring course, and then using Clingo’s soft constraints in order to find the course that is closer to the user’s own learning style scores. For more information on the tutoring course selection, see section 4.6.3

The projection program is responsible for calculating Afflog’s new state, given the old state and the current action. This is the most commonly used Answer Set program as it is run by the control method in the tutor layer, for each action. This program is also responsible for marking the taught tutorials of a chapter according to how the user fared on the corresponding test as described in section 4.1. For more information on the tutoring course selection, see section 4.6.3.

Finally, the "Replan tutoring courses" program, which we will be referring to as the "replanner", has the same functionality as the planner, with the difference that it does not start selecting actions from the beginning of the course, but from a given input state. Here, we use the term "replanning" loosely, and not as the deterministic planning community, as we do not use algorithms such as D*, [78] to repair previous plans, but instead plan again from scratch, by changing the input state. However, since the input state is the product of a previous plan, replanner is used when the user has failed a test, and consequently the control method must choose different tutorials for the corresponding chapter of that test. For more information on the tutoring course selection, see section 4.6.2.

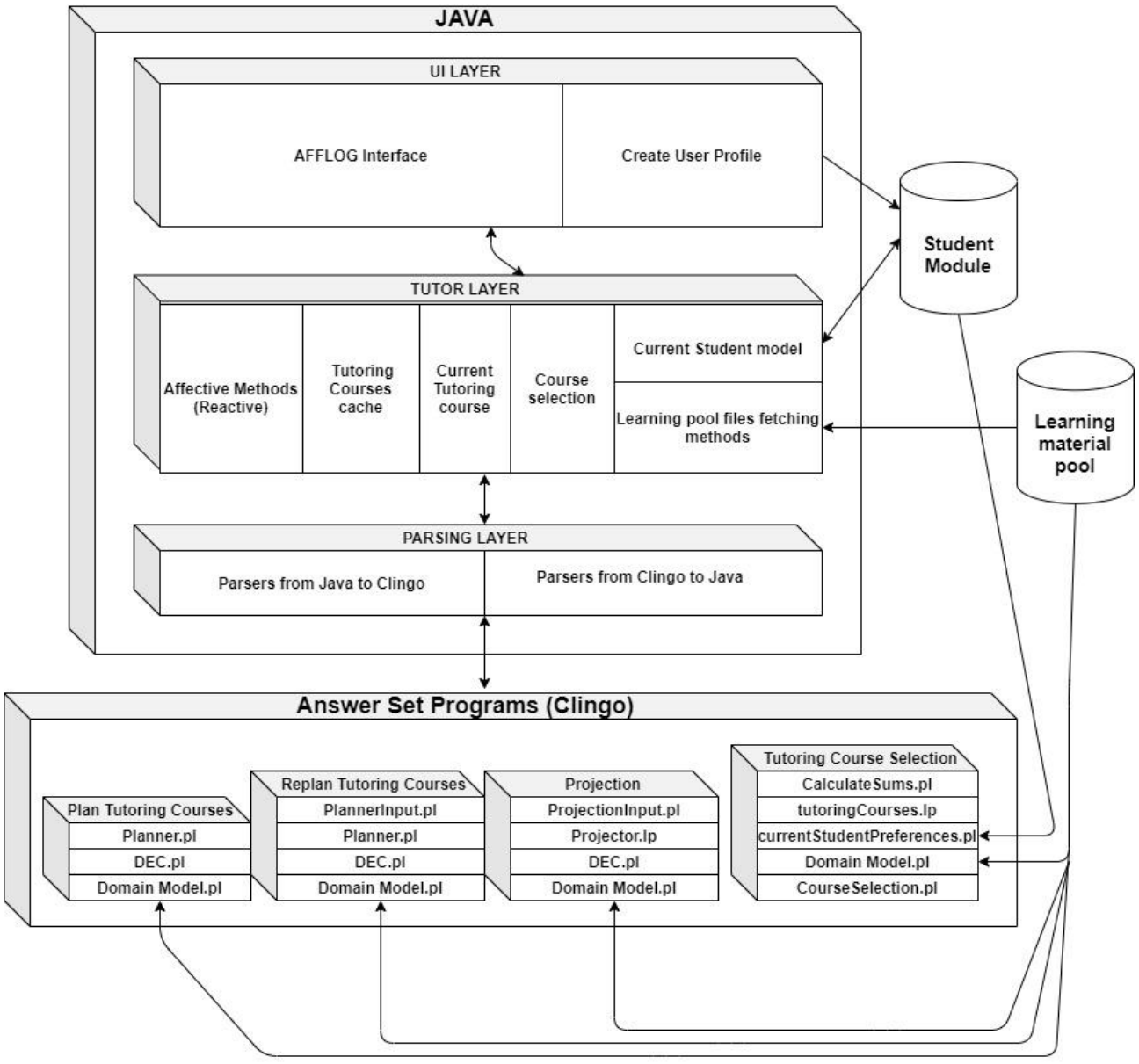

Figure 4.4. AFFLOGs Architecture. The arrows denote the flow of the data.

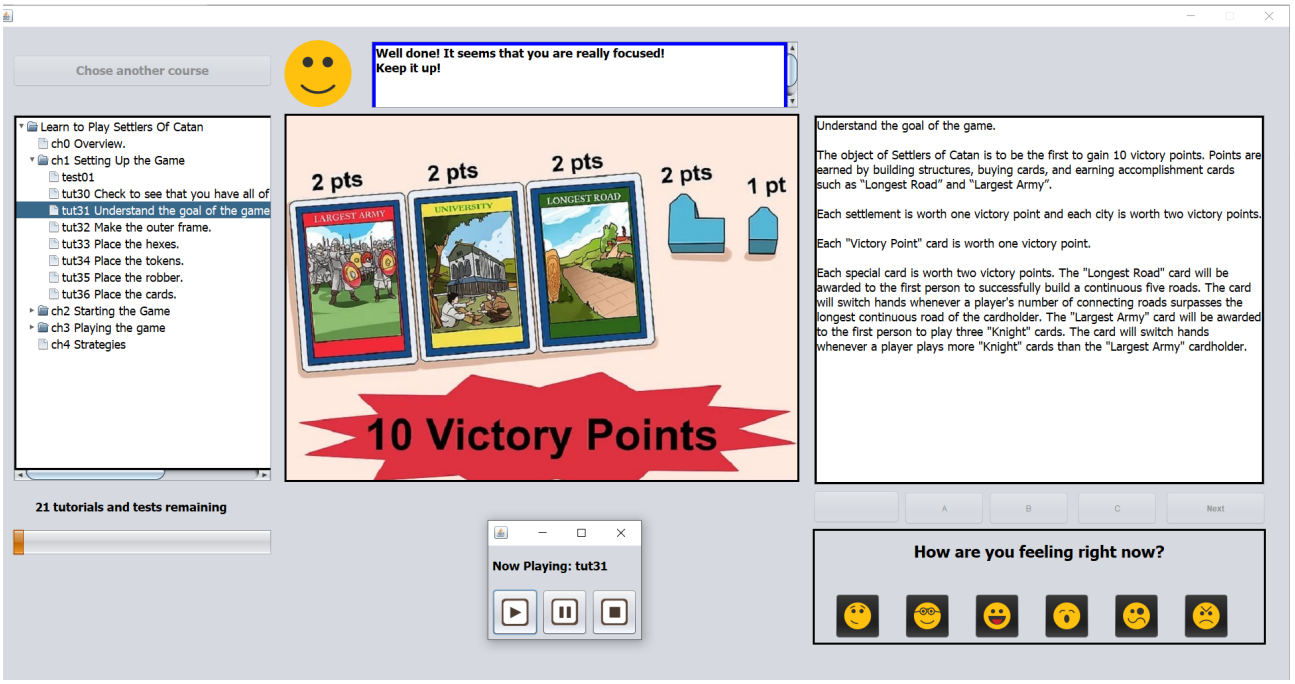

Figure 4.5 : AFFLOGs User Interface.

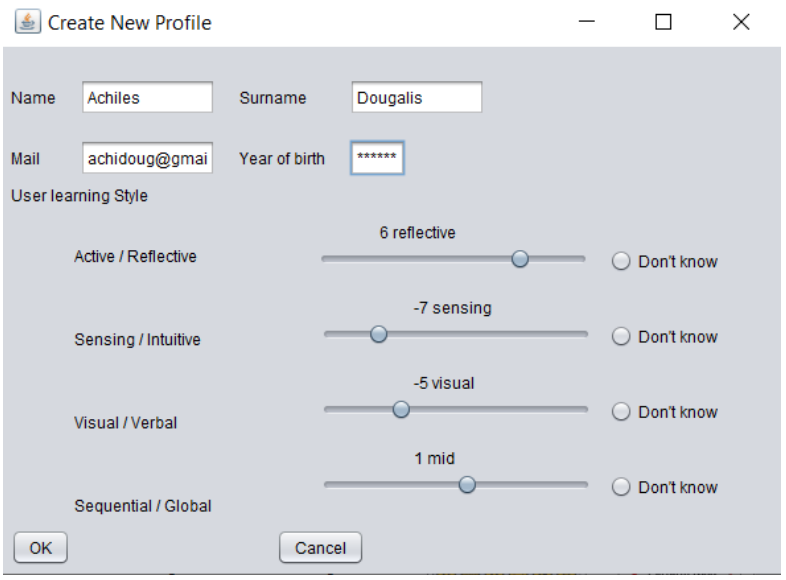

Figure 4.6: Profile Input.

4.6 Properties

In this section we list the predicates and properties of the domain model file used by AffLog in the Answer Set programming language of Clingo. We also attempt to explain the main syntax rules that

are used. For more information on Clingo and the ASP rule syntax in general the reader should consult [2, 5]. In the Clingo language, we represent predicates according to their arity. If a predicate **p** has arity **x**, then we refer to it as **p/x**. A unary predicate is a relation with one constant or variable while a binary predicate is a relation between two constants or variables. The domain model is composed of the following unary and binary predicates:

chapter/1, section/1,tutorial/1 test/1 represent respectively, all the chapters, sections, tutorials and tests that a domain file contains. For example, in the domain file of the "Settlers of Catan" course, we have four **chapter/1** predicates: **chapter(ch1)**, **chapter(ch2)**, **chapter(ch3)**, and **chapter(ch4)**. Here **ch1**, **ch2**, **ch3** and **ch4** are constants representing the four chapters of a tutoring course. In Clingo, we can use the ";" character as syntactic sugar allowing us to write these predicates as: **chapter(ch1;ch2;ch3;ch4).** Note that the symbol "." represents the end of a logical statement.

The predicates **tutorialtype/1**, **tutorialDifficulty/1** and **tutorialSpeed/1** contain all the possible values that can characterize a tutorial in terms of media type, difficulty and tutorial speed. These values may be used for a more detailed tutorial selection. These values are:

```
tutorialtype(text;picture;audio;video).

tutorialDifficulty(easy;normal;hard).

tutorialSpeed(slow;normal;fast).
```

hasChapterSection/2 ,**chapterBegins/2**, **chapterEnds/2**, are binary predicates that represent the relations between a chapter and a section, while **hasChapterTest/2** states which tests belong to what chapter. We use the "**,**" to separate each constant or variable in a relation. For example, chapter **ch1** in the domain file of the "Settlers of Catan" belongs to the following statements:

```
hasChapterSection(ch1,start).        hasChapterSection(ch1,sect2).
hasChapterSection(ch1,sect3).        hasChapterSection(ch1,sect4).
hasChapterSection(ch1,sect5).        hasChapterSection(ch1,sect6).
hasChapterSection(ch1,sect7).

chapterBegins(ch1,start). chapterEnds(ch1,sect8).

hasChapterTest(ch1,test01). hasChapterTest(ch1,test02).
```

Here, constants **start,sect2,sect3,sect4,sect5,sect6,sect7,sect8** belong to the **section/1** relation, and predicates **test01,test02** belong to the **test/1** relation.

For each tutorial in the domain file we have the following relations: **hasStart/2** **hasEnd/2**,**hasType/2**, **hasactiveReflective/2**, **hasSensingIntuitive/2**, **hasVisualVerbal/2**, **hasSequentialGlobal/2**, **hasDifficulty/2**, and **hasSpeed/2**.

The relations **hasStart/2** and **hasEnd/2** represent in what section does a tutorial start and end respectively. The relations **hasActiveReflective/2**, **hasSensingIntuitive/2**, **hasVisualVerbal/2**, and **hasSequentialGlobal/2** characterize a tutorial's standing regarding the respective Felder and Silverman learning style model dimensions, by using the values 1,2 and 3 in the second field. For example the predicates **hasActiveReflective(x,1)** **hasSensingIntuitive(x,2)** and **hasVisualVerbal(x,3)** state that tutorial **x** is suitable for active users, not particularly suitable or unsuitable for sensing or intuitive users, and suitable for verbal users. **hasDifficulty/2**, and **hasSpeed/2** are relations between a tutorial and the constants of the **hastype/1** and **hasDifficulty/1** relations. For example, tutorial **tut01** in the domain file of the "Settlers of Catan" is part of the following statements:

```
hasStart(tut01,start).  hasEnd(tut01,sect2).  hasType(tut01,text).
hasType(tut01,picture).              hasActiveReflective(tut01,1).
hasSensingIntuitive(tut01,1).          hasVisualVerbal(tut01,2).
hasSequentialGlobal(tut01,1).        hasDifficulty(tut01,easy).
hasSpeed(tut01,fast).
```

Using the above predicates we can construct rules, using logical entailment. Rules in ASP consist of the rule head, followed by the symbol **:-** and the body of the rule. the argument at the head of the rule is true, only if the argument of the body is also true. For example, rule **(1)** below states that if a chapter **CH2** begins where another chapter **CH1** ends (at the section **SECT)**, then the chapter **CH1** will be before **CH2**. Note that **CH1** and **CH2** are *variables* and not *constants*, and thus, before the program containing this rule is solved, it has to be grounded. Grounding means that all variables will be instantiated to constants found in these relations. So, after the program containing rule (1) is grounded, we will have the relations found in (2).

```
chapterBefore(CH1,CH2):-chapterBegins(CH2,SECT);
chapterEnds(CH1,SECT).                                    (1)

chapterBefore(ch1,ch2).chapterBefore(ch2,ch3).
chapterBefore(ch3,ch4).                                   (2)
```

We can also use entailment with the same relation multiple times as in (3) and (4) in order to get the relations in (5):

```
chapterBefore(CH1,CH3):-chapterBefore(CH1,CH2);
chapterBefore(CH2,CH3).                                   (3)

chapterBefore(CH1,CH4):chapterBefore(CH1,CH2);
chapterBefore(CH2,CH3); chapterBefore(CH3,CH4).           (4)

chapterBefore(ch1,ch3).chapterBefore(ch1,ch4).
chapterBefore(ch2,ch4).                                   (5)
```

Finally, we can create more complex relations such as (6). The **containsChapter(x,chx)** will be true, if the chapter **chx** is between the chapters where the tutorial **x** starts and ends. Note

that the ";" symbol between each predicate in the body of the rule represents a logical conjunction. This means that the head of the rule will be true, only if all the predicates at the body are also true.

```
containsChapter(TUT,CH):- hasTutorialStart(TUT,SECT1);
hasChapterSection(CH1,SECT1); hasTutorialEnd(TUT,SECT2);
hasChapterSection(CH2,SECT2); chapterBefore(CH1,CH);
chapterBefore(CH,CH2); not chapterEnds(CH,SECT2);
chapter(CH;CH1;CH2).                                    (6)
```

4.7 ASP programs

In this section we dive deeper into the four Answer Set programs in order to give to the reader a measure of understanding of their usage. As seen in figure 4.3, all these programs make use of the domain model file that we described in the previous section. Another file that is used in three of these programs is the Discrete Event Calculus (DEC.lp) file that will be discussed in detail.

4.7.1 DEC.lp

As seen in chapter 2, the Event Calculus is an action language that reasons with events and fluents, in a commonsense world. In the Discrete Event Calculus (DEC), there are discrete time points. For the purposes of AffLog, we used the Kim et al. translation of DEC, [6], that is given in the f2lp System (http://reasoning.eas.asu.edu/f2lp/). We translated it to the latest version of Gringo as the #domain predicates are obsolete. Also, for the purposes of our work, we simplified excluded ReleasedAt predicates from the axiomatization, keeping only four of the DEC axioms.

In the DEC.lp file we represent the discrete time points as a predicate **`time/1`**. This predicate is instantiated as: **`time(0..maxstep)`** which denotes that there are going to be **`Maxstep+1 time`** predicates where **`maxstep`** is a constant that is defined when the ASP program runs.

At any given time point, a number of fluents may be true. We say that any fluent that is true at a certain timepoint, *holds* at this timepoint. This is described in ASP as:

```
{holdsAt(F,T)}:-fluent(F);  time(T).
```

To describe an event in ASP happening at a time point **`T`**, we use the predicate **`happens`**:

```
happens(E,T):- event(E); time(T).
```

An event that happens at a timepoint **`T`**, may terminate a certain predicate:

```
terminates(happens(E,T),holdsAt(F,T)):-  event(E);  fluent(F);
time(T).
```

Also, an event that happens at a timepoint **`T`**, may initiate a certain predicate:

```
initiates(happens(E,T),holdsAt(F,T)):-  event(E);  fluent(F);
time(T).
```

A fluent that holds in the commonsense world, will hold forever until it is terminated by an event. This is also known as the commonsense law of inertia. To describe that we use the following two rules:

```
holdsAt(F,T+1) :- holdsAt(F,T); not _new_pred_1(T,F); time(T);
fluent(F); T<maxstep.                                          (7)

_new_pred_1(T,F) :- happens(E,T); terminates(E,F,T); event(E);
fluent(F).                                                     (8)
```

The rules (7) and (8) state that a fluent that holds at timepoint **T**, will hold at timepoint **T+1**, as long as an event won't terminate this fluent at timepoint **T**.

Rule (9) states that a fluent **F** at **T+1** will hold if it is initiated by an event at timepoint **T**.

```
holdsAt(F,T+1):-happens(E,T),initiates(E,F,T),event(E),fluent(F),
time(T),T<maxstep.                                             (9)
```

Rule (10) states that a fluent that is terminated at timepoint **T** cannot hold at timepoint **T+1**.

```
:- {not holdsAt(F,T+1)}0 ;happens(E,T); event(E); fluent(F);
terminates(E,F,T);time(T);T<maxstep.                           (10)
```

Finally, rules (11) and (12) state that when a fluent is initiated at timepoint **T** it has to hold at timepoint **T+1**.

```
 :- {not holdsAt(F,T+1)}0; not holdsAt(F,T);not _new_pred_2(T,F);
fluent(F); time(T); T<maxstep.                                 (11)

_new_pred_2(T,F) :- happens(E,T);    initiates(E,F,T); event(E);
fluent(F).                                                     (12)
```

Rules (11) and (12) are "headless" rules known in ASP as constraints. Constraints are rules without a head, and are used to restrict the search space of the ASP solver.

4.7.2 The planner

As seen in figure 4.3, the : "Plan tutoring courses program" also known as the planner, consists of the DEC.lp and Domain Model.lp files as well as the planner.lp file. In this subsection we explain the main ASP rules of the planner, expanding the 4.4 section. First we use the domain model predicates to build events and fluents.

```
fluent(taught(TUT,C)):-tutorial(TUT); cognitiveState(C).

fluent(didTest(TST,TR)):-test(TST);testResult(TR).

event(doTutorial(TUT)):- tutorial(TUT).
```

```
event(doTest(TST,TR)):- test(TST);testResult(TR).
```

Rules 13 - 16, are the most essential parts of the planner. Rule 13 lets all possible events happen at any time while 14-16 constrain those events. Specifically, while rule 14 states that only one event may happen at most at a certain timepoint, rule 15 ensures that at least one event will happen at a certain timepoint. Finally, rule 16 states that no wrong doTest events shall happen, thus ensuring that, for the purposes of this planner, all tests are going to be answered correctly. To sum up, rules 13-16 dictate that at every timepoint, one event will happen, and if this event is a doTest, its result will be "correct".

```
{happens(E,T)}:-event(E);time(T).                                    (13)

:- happens(E1,T); happens(E2,T); E1 !=E2; event(E1); event(E2);
time(T).                                                             (14)

:-not happens(_,T);T!=maxstep;time(T).                               (15)

:-happens(doTest(TST,wrong),T);time(T);test(TST).                    (16)
```

Now that we have established what events may happen, we need to decide the starting conditions, as well as the planner's terminal conditions. As a starting condition, that is at timepoint 0, the only fluent that will hold is the **absenceofTutorials** tutorial. This tutorial is not real but is used in order to represent that no other tutorials hold when it holds.

```
holdsAt(taught(absenceofTutorials,unknown),0).
```

We also need to make sure that not any other fluents except the absence of tutorials will be holding at timepoint 0. This is done by constraints 17 and 18.

```
:-holdsAt(taught(TUT,C),0);cognitiveState(C);
TUT!=absenceofTutorials.                                             (17)

:-holdsAt(didTest(TST,TR),0).                                        (18)
```

To establish the terminal condition we define the goal state. With rule 19, we define that the goal/1 predicate will be true whether a tutorial that ends at the end of chapter 3, that is, section 20 is correct. We could alter this condition by changing the **hasTutorialEnd** predicate to point to another section, for example, the last section of chapter 4.

Rule 20 ensures that no taught tutorial will be unknown if the **goal** predicate is true. This ensures that the resulting set of tutoring courses will contain a test for each chapter, as this is the only way to have correct tutorials. Finally, rule 21 makes sure that the **goal** predicate will be true at the final time point.

```
goal(T) :- holdsAt(taught(TUT,correct),T);
hasTutorialEnd(TUT,sect20); tutorial(TUT); time(T).                  (19)

:-goal(T);holdsAt(taught(TUT,unknown),T).                            (20)
```

```
goal(maxstep).                                                    (21)
```

Now that we have established how the events will happen, and the initial and terminal conditions of the planner, the only thing left to do is to decide what fluents will be initiated, as well as what constraints are needed in order to ensure that the planner works correctly. In order to help the reader understand the main ideas behind the planner, we will focus on the simpler **initiates** and **terminates** rules as well as some of the constraints. The rest of the rules will be available in the appendix of this dissertation.

Rule 22, states that the event **doTutorial** will initiate the respective **taught** fluent, with the cognitive value **unknown,** since a test is needed to verify if the user has understood that tutorial. The only **terminates** rule that a **doTutorial** can do is rule 23, that states that when any **doTutorial** action happens, then the **taught(absenceofTutorials,unknown),T)** will be terminated.

```
initiates(doTutorial(TUT),taught(TUT,unknown),T):-
                                        tutorial(TUT);time(T).
                                                          (22)
```

```
terminates(doTutorial(TUT),taught(absenceofTutorials,unknown),T):-
tutorial(TUT);time(T).
(23)
```

Rules 24 and 25, are identical in the sense that their bodies are the same, and they both describe what will happen after the event **doTest(TST,correct)** happens. In the first rule, all the unknown tutorial fluents that belong to the same chapter as the test will terminate, and in the second rule, the same tutorials will initiate, only this time, as correct tutorials.

```
terminates(doTest(TST,correct),taught(TUT,unknown),T):-
holdsAt(taught(TUT,unknown),T); hasChapterTest(CH,TST);
hasTutorialEnd(TUT,SECTEND); hasTutorialStart(TUT,SECTSTART);
hasChapterSection(CH,SECTSTART);
1{tutorialEndsAtEndOfChapter(TUT,CH);
hasChapterSection(CH,SECTEND)}; time(T).                          (24)
```

```
initiates(doTest(TST,correct),taught(TUT,correct),T):-
holdsAt(taught(TUT,unknown),T); hasChapterTest(CH,TST);
hasTutorialEnd(TUT,SECTEND); hasTutorialStart(TUT,SECTSTART);
hasChapterSection(CH,SECTSTART);
1{tutorialEndsAtEndOfChapter(TUT,CH);hasChapterSection(CH,SECTEND)
}; time(T).                                                       (25)
```

The**1{tutorialEndsAtEndOfChapter(TUT,CH);hasChapterSection(CH,SECTEN D) };**part of both rules is known as a cardinality constraint in ASP. The 1 on the left of the left angle bracket denotes that at least 1 statement inside the brackets must be true. This allows us to

create logical disjunctions meaning that the rule will fire when either one of the two predicates is true.

The constraint rule 26, states that a tutoring course cannot have two or more tutorials that start at the same section. Rule 27 states that the same tutorial cannot hold with 2 or 3 different cognitive states. Finally rule 28 states that there cannot be taught tutorials of a chapter, if there are unknown tutorials of previous chapters.

```
:-happens(doTutorial(TUT1),T);holdsAt(taught(TUT2,unknown),T);
hasTutorialStart(TUT1,SECT);hasTutorialStart(TUT2,SECT);
TUT1!=TUT2.                                                    (26)

:-holdsAt(taught(TUT,C),T);holdsAt(taught(TUT,C1),T);C1!=C.    (27)

:-happens(doTutorial(TUT),T);holdsAt(taught(TUT1,unknown),T);
hasTutorialEnd(TUT,SEC);hasTutorialEnd(TUT1,SEC1);
hasChapterSection(CH,SEC); hasChapterSection(CH1,SEC1);
transchapterBefore(CH,CH1);time(T).                            (28)
```

When running the planner, we have to input the value of the maxstep variable. This is done by first finding the minimum and the maximum value of the maxstep value from the domain file and the goal, and then running the planner sequentially starting from the minimum maxstep value, and incrementing it by one, until it reaches the maximum value. To calculate the minimum and maximum values, we need to know how many chapters there are until the goal section in order to find the number of tests that we need to include in the resulting tutoring courses, and what is the minimum and maximum number of tutorials that are needed to reach the goal section. For example, the “settlers of Catan” domain file has 4 chapters, and can be traversed by 2 tutorials at the minimum, and 21 tutorials at the maximum. That gives us a minimum maxstep value of 6, and a maximum maxstep value of 25. Subsequently, in order to plan all the tutoring courses for The “settlers of Catan” domain we need to run the planner, for maxstep=6, and then run it again multiple times, each time incrementing maxstep by 1, before running the planner one last time for maxstep=25.

The “Replan tutoring courses" program is identical to the planner, with the only difference being the plannerInput.pl which holds Afflog’s current state. For example, the ASP code below describes the input of the “Replan tutoring courses' ' program stating that at timepoint 0 the only fluents that are true are the tests **`test01, test04, test06`** and **`tut21.`**

```
:-holdsAt(taught(TUT,C),0);cognitiveState(C);TUT!=tut21.

:-holdsAt(didTest(TST,TR),0);test(TST);TST!=test06;TST!=test04;
TST!=test01.

holdsAt(didTest(test06,wrong),0).

holdsAt(didTest(test04,correct),0).

holdsAt(taught(tut21,wrong),0).
```

```
holdsAt(didTest(test01,correct),0).
```

4.7.3 Projection

The projection program is responsible for finding the new state of the system, given the old state, and the current action. Its rules are very similar to the planner program as it uses the same DEC fluents and actions, but since we are interested in the state after a single action, we only need to run the program once, for maxstep=1. Also, the projection program keeps track of the user's emotional state, so in addition to the predicates of the planner we also have the following predicates:

```
emotionalState(neutral;flow;delight;bored;confused;frustrated).

emotionArousal(high;mid;low).

fluent(currentEmotion(E,A))  :-
emotionalState(E);emotionArousal(A).
event(userExperiencesEmotion(E)) :- emotionalState(E).

initiates(userExperiencesEmotion(E),currentEmotion(E,high),T) :-
holdsAt(currentEmotion(E,mid),T);time(T); emotionalState(E).

terminates(userExperiencesEmotion(E),currentEmotion(E,mid),T)
:-holdsAt(currentEmotion(E,mid),T);time(T);emotionalState(E).

initiates(userExperiencesEmotion(E),currentEmotion(E,mid),T) :-
holdsAt(currentEmotion(E,low),T);time(T); emotionalState(E).


terminates(userExperiencesEmotion(E),currentEmotion(E,low),T)
:-holdsAt(currentEmotion(E,low),T);time(T);emotionalState(E).

initiates(userExperiencesEmotion(E),currentEmotion(E,low),T) :-
holdsAt(currentEmotion(E1,A),T);emotionArousal(A); not E1=E;
emotionalState(E);emotionalState(E1); time(T).

terminates(userExperiencesEmotion(E),currentEmotion(E1,A),T) :-
holdsAt(currentEmotion(E1,A),T); not E1=E; emotionArousal(A);
     emotionalState(E); emotionalState(El); time(T).
```

The emotional state of the user is the **currentEmotion** fluent that contains the dominant emotion of the user, and its respective arousal level. The first time a user experiences an emotion, arousal will be low. If the same emotion persists, then in the next projection output, the arousal level of the fluent will be mid, and in the subsequent projections the arousal of the fluent will be high. Afflog's affective responses discussed in section 4.3 are taken by consulting the current state of this fluent.

4.7.4 Tutoring Course Selection

The tutoring course selection program has basically two programs. The first program, sums every learning style value of each tutorial for each tutoring course and creates an average score of every learning style for all tutoring courses. The second program uses Clingo's soft constraints, [5], in order to select the tutoring course that better fits the learning style of the current user.

First, we sum all the learning style values of each tutorial for each tutoring course using Clingo's #sum aggregate function. For example, rule 29, sums all **hasTutorialactiveDimension/2** values of the course **C**, creating the predicate **courseActiveVal/2.** Since there are tutoring courses containing only 1 tutorial, and others containing up to 26 tutorials, we would like to normalize these values so that we could compare tutoring courses of different tutorial numbers. To do this, we first find the number of tutorials per course with rule 30, and then we divide the learning style values by that number as shown in rule 31. Note that we have also multiplied the normalized value by 100 since clingo does not support non-integer numbers. Finally, we sum all **normalizedCourseActiveVal/2** predicates, dividing them by the number of all the tutoring courses in order to create the **avgCourseActiveVal/1** predicate.

```
courseActiveVal(C, S) :- S = #sum{ I,TUT:courseContains(C,TUT),
hasTutorialactiveDimension(TUT,I)},course(C).                    (29)

tutorialsPerCourse(C, S) :- S = #count {T: courseContains(C,T),
tutorial(T)}, course(C).                                         (30)

normalizedCourseActiveVal(C,S):-S=#sum{T*100/SIZE:
tutorialsPerCourse(C,SIZE),courseActiveVal(C, T)}, course(C). (31)
```

In the second program, we create weak constraints for each learning style score of the current user in order to select the tutoring course that better fits to the learning style of the user. For example, rule 32 states that in order for **b** to hold for a tutoring course the **normalizedCourseActiveVal/2** value of the tutoring course must be greater than the **avgCourseActiveVal/1** value, since the user is active and not reflective. Rule 33, states that a candidate course that satisfies **b**, gains 5 points. The optimal course for the user, is the one that manages to accumulate more points, and thus, the one that fulfills most of the user's learning style preferences.

```
b:-course(C);S>AVG;userActiveReflective(1);
normalizedCourseActiveVal(C,S);avgCourseActiveVal(AVG).          (32)

:~b.[5]                                                          (33)
```

4.8 Usage

In this section, we present how AffLog works by showcasing two examples of use during a certain time period of a session, in the hope that the reader will acquire a better insight into the system's workings. The first example describes how the session starts, and the second describes what happens when a user fails a test.

4.8.1 Starting the Session

As seen in figure 4.7, when the session starts, most of the functionality of the interface is disabled, except for a window that allows the user to chose his/her profile and whether he/she wants a basic or an advanced course, and the "Create Courses" button at the upper corner of the interface. After the user chooses his/her profile and decides whether to choose a basic or an advanced tutoring course, he/she presses the "Create Courses" button at the upper left corner of the screen.

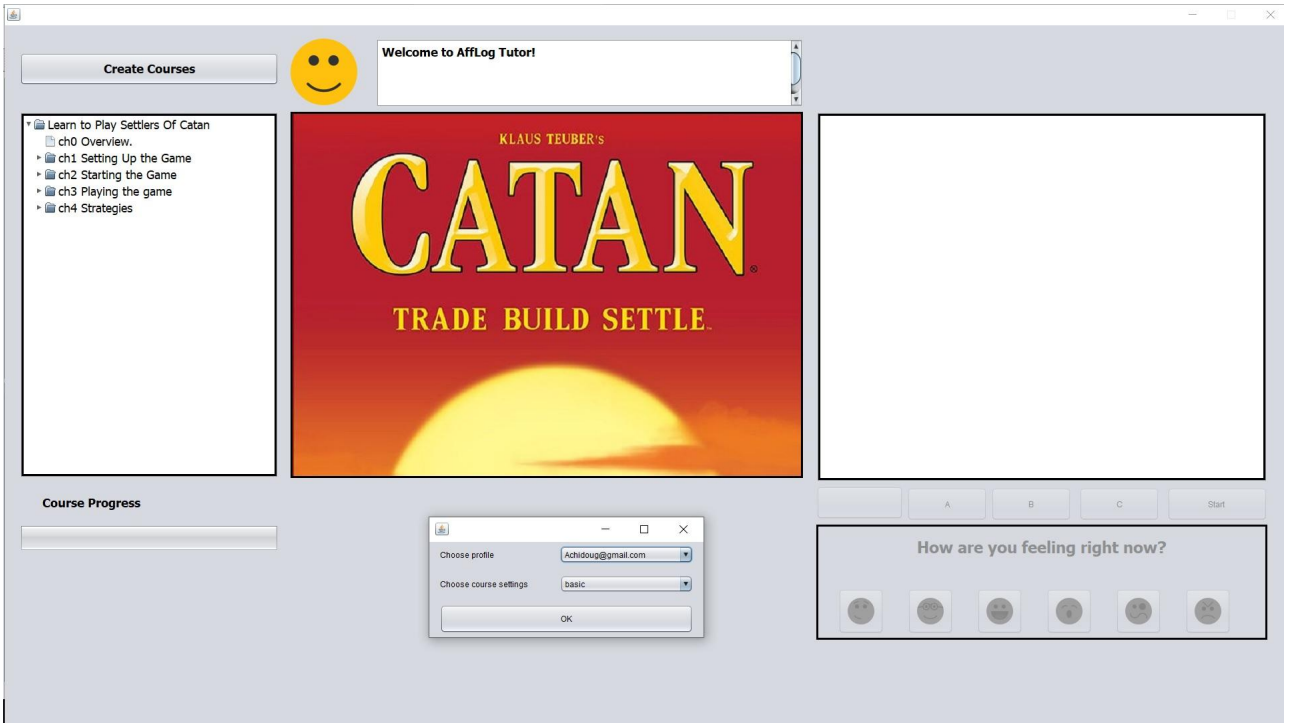

Figure 4.7: Starting the session.

The system then runs the planner and "tutoring course selection" ASP programs and presents the selected tutoring course to the user, listing the number of tutorials and tests as well as any warnings regarding the tutorials. For example, in figure 4.8 we see that AffLog has found a tutoring course with 19 tutorials and 3 tests that matches the learning style of the user except for the sensing/intuitive dimension where the system could not find a tutoring course with a sensing score greater than average. At this point, the user can choose to carry on and start the selected tutoring course by pressing the "next" button on the right, or choose another tutoring course by pressing the button at the upper left corner of the screen, now renamed "choose another course". If the latter is pressed, then AffLog will run "tutoring course selection" again, this time excluding the course that was previously selected.

When the user presses the “next” button, the first tutorial will be presented in the main window (figure 4.9), and the file directory on the left will fill with all the tutorials in this tutoring course. When the user has read or watched the tutorial, he/she can press one of the now-enabled emotional buttons at the bottom right corner of the UI in order to go to the next tutorial.

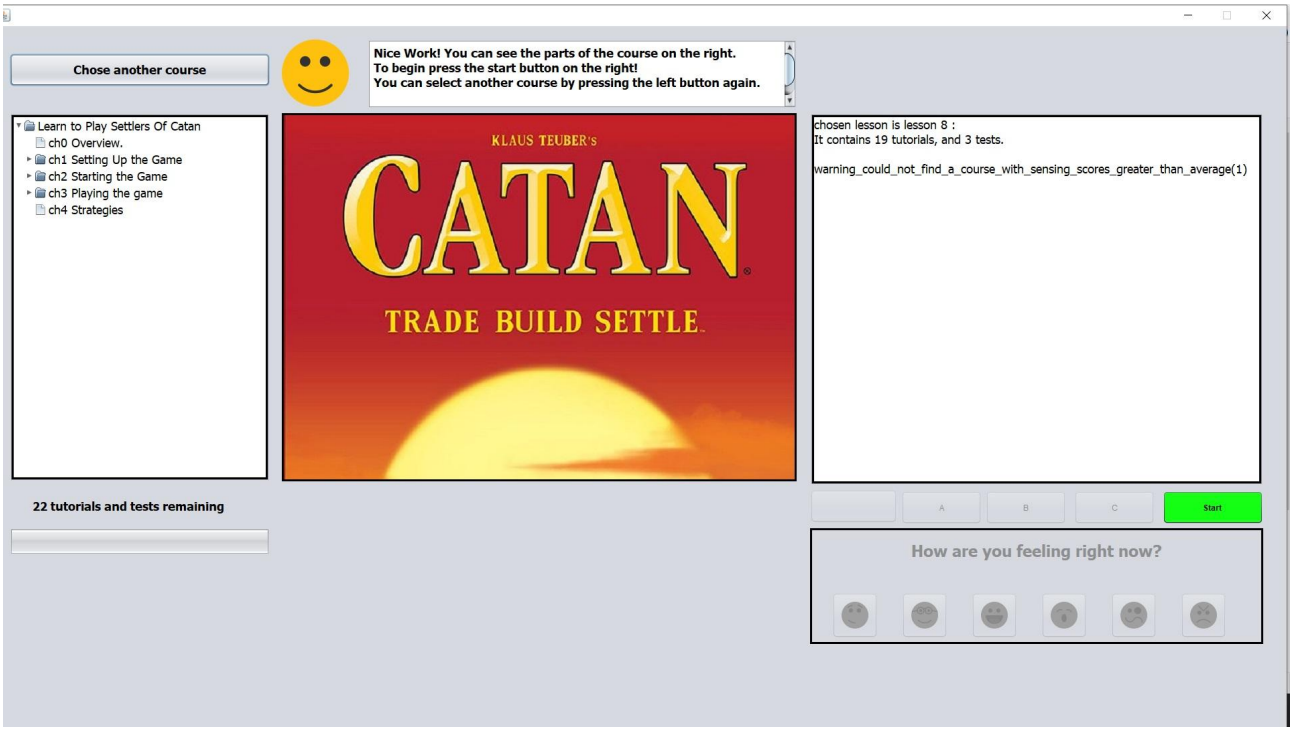

Figure 4.8: Select a tutoring course.

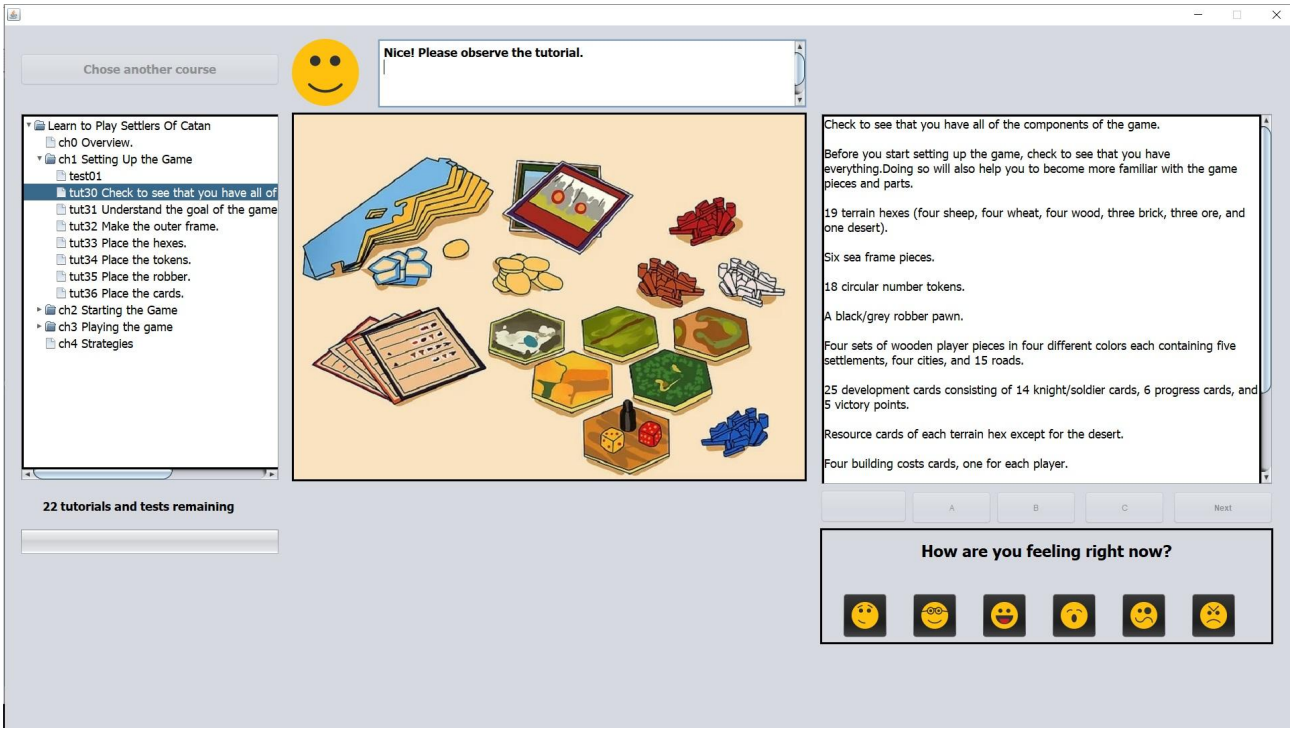

Figure 4.9: The first tutorial is presented**.**

After pressing the appropriate emotion, the user will be directed to the next tutorial (figure 4.10). In this figure, we can see the affective reaction of the tutor after the user pressed the “flow” button, at the upper center part of the UI. Also note the tutorial bar on the left bottom corner, that shows how many tutorials and tests remain in order to complete the tutoring course.

Figure 4.10: The second tutorial is presented.

4.8.2 Failing and passing a test

In this example, we examine what happens when the user has to answer a test. In figure 4.11, the user has completed all the tutorials of the second chapter, and he/she is presented with a test. The test is a multiple choice question about a fact contained in the tutorials of the second chapter. Unfortunately the user selects a wrong answer (figure 4.12), and the tutor informs him/her of the mistake he/she made, as well as the correct answer and the reasoning behind it. When the user finishes reviewing the correct answer, he/she presses the “next” button.

Afflog then performs replanning and tries to find a new tutoring course, keeping the same tutorials before chapter 2, but choosing different tutorials than those presented in chapter 2, if they exist. Of course, even if replanning chooses the same tutorials for chapter 2, it is probable that the user will have the opportunity to revisit those tutorials and hopefully, pay more attention to them this time.

After the replanning, AffLog restarts chapter 2, and presents the tutorials of the new tutoring course to the user. After he/she observes them, the system will present him/her with a different test (figure 4.13).

Figure 4.11: The user must select the right answer to pass the test.

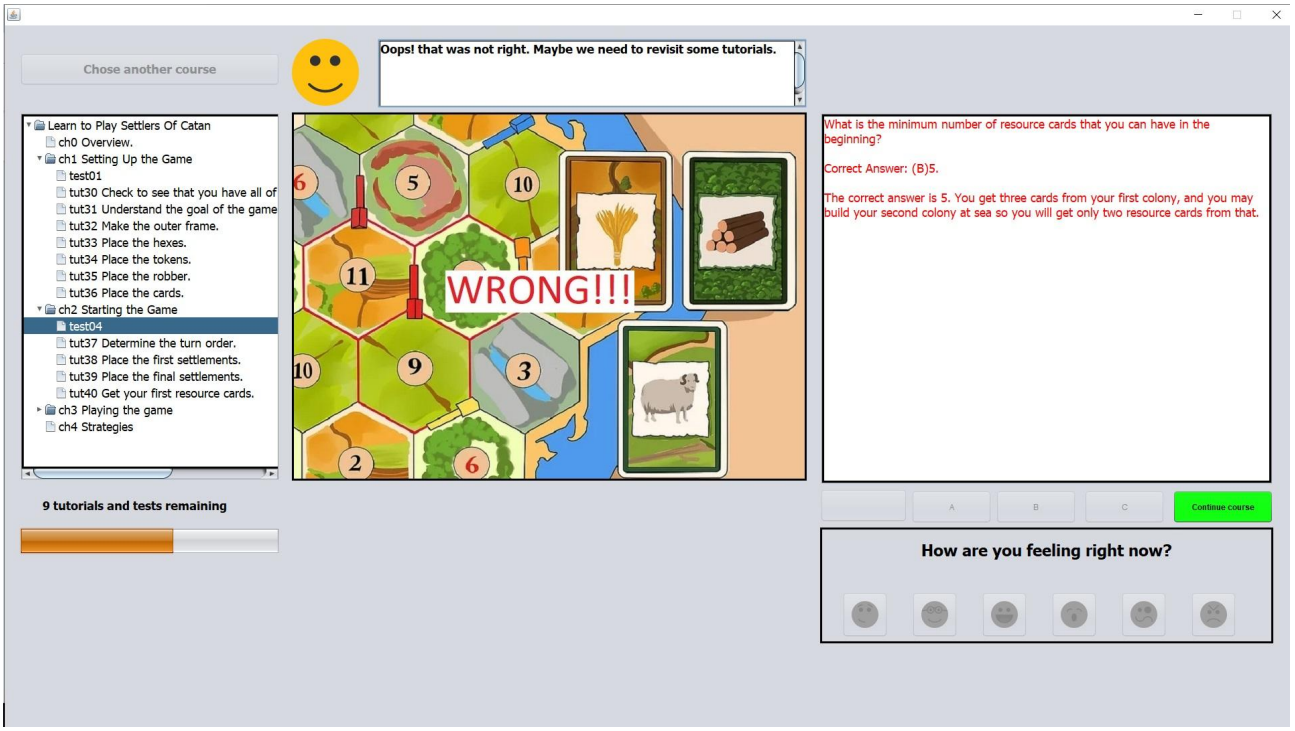

Figure 4.12: The user answers incorrectly.

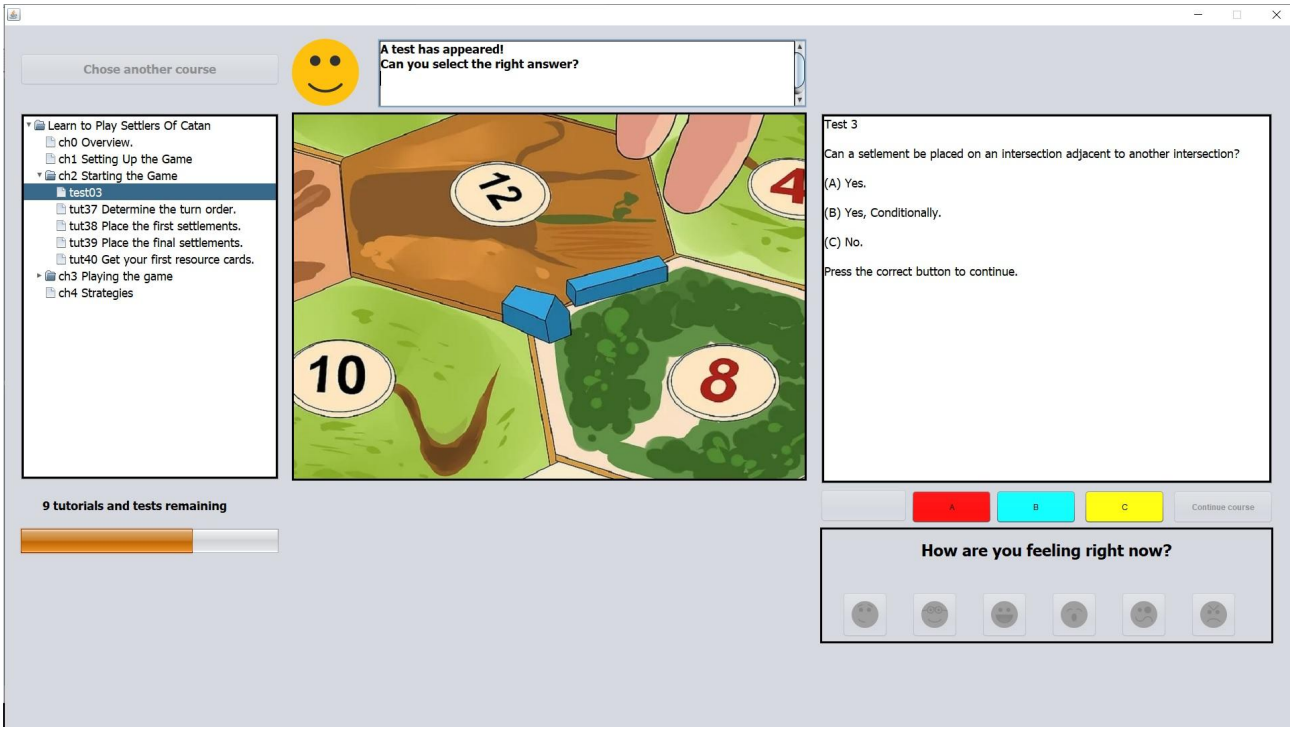

Figure 4.13: New test is presented.

4.9 Limitations

The main limitations of the Afflog System are related to the kind of courses that someone can create. Specifically, courses with a small number of tutorials, or with less than 2 tutorials covering each subsection will not be able to sufficiently adapt to the Learning Styles of all possible users.

On the other hand, large courses with more than 40 tutorials and 2 or more tutorials for each subsection will exponentially increase the possible courses and the time needed for the planner to calculate them. Specifically, a course with 48 tutorials and 7 tests with 2 or more tutorials for each subsection, will create more than 8000 possible courses. With the initial planning time being more than 5 minutes in an average computer. Of course, initial planning and course selection could be easily done “offline” before any interaction with the user takes place, as long as that user’s learning preferences are known. However, there is still course replanning resulting from a failed test to consider. This can result in a number of possible courses less or equal than that of the initial planning while such replanning has to be done during user interaction that can lead to the user waiting for minutes for an alternative course. Note here, that this is an implementation problem and not a design or an ASP problem. A Planner using all optimizations available to Clingo such as incremental grounding would fare much better.

Other limitations include the use of the Felder and Silverman Learning style model and not other learning styles, and that the current Emotional recognition of the system is not connected to any affect-sensing mechanism but is being done by the user or by an expert monitoring him/her.

Chapter 5
The AffLog RL System

In this chapter the design and implementation of the AffLog RL system are presented. First we offer some insight on why we created a RL version of Afflog. Then we present key differences between the two systems. In the next section we describe how we turned the affective tutoring system problem, to a Reinforcement Learning problem. Then an overview of the system's functionality and architecture is presented. After that we present a section offering a detailed explanation of the methods and Answer Set Programs that were used.

5.1 Motivation

After the creation of the Afflog system, discussions were held regarding the operation and the principles behind the system. These discussions produced the following arguments:

Although Learning styles was a way to select appropriate tutorials, and create a course, there is a lack of scientific evidence to justify learning style theories as we mentioned in Chapter 1. Considering this, is the construction of a course where each tutorial corresponds to the specific learning style of the user optimal? Or, for example, would it be better to only have the majority of the tutorials correspond to the learning style of the user and the rest correspond to a different style? Also, considering that each person has evidently unique ways of learning, a clustering method based on that person's learning style feels simplistic and does not seem to offer the best adaptation possible to the user's needs. We argue that a tutoring agent that learns from experience, given enough time, will be able to respond to the learning preferences of the user more accurately than a learning style strategy.

Another argument is that the constant affective feedback from the system could overstimulate the user, and thus impair the learning process. Instead the system should try to offer affective feedback sparingly, in order to increase the chances of assisting the user. An adequately trained RL tutor could easily identify critical tutorials, as these tutorials, when presented to the user, would yield a greater reward than other tutorials. It is not clear how one could make use of this information regarding affective feedback. Maybe it would be prudent not to disturb the user with affective feedback when he/she is presented with a critical tutorial, or a series of critical tutorials so as not to disturb the user's train of thought. Then again, maybe that is exactly when affective feedback is

needed as it is critical to keep the user motivated. So what is the best choice? The trained RL tutor would know the answer based on the rewards yielded by the sequences of tutorials, tests and affective actions.

Regarding tutorials and tests, Afflog cannot discriminate between two tutorials with the same learning style properties that cover the same sections. Let us call a pair of such tutorials *equivalent* tutorials. This is also true for sequences of tutorials with the same properties, and also tests that cover the same knowledge. So there can be equivalent sequences, and equivalent tests. These differences may be different sentences that describe the same subject, different pictures, or different exercises taken from the same chapter. As long as they have the same properties (those described in 4.5), Afflog will not discriminate between equivalent tutorials or tests when planning a course for the user, but will create a set of courses that correspond to the learning style of the user and then randomly select one course. The user however, can notice the difference between two equivalent tutorials or tests and may have a preference. We believe that a trained RL tutor should be able to present to the user a tutorial or test that is better suited to the user than its equivalent.

5.2 Advantages of AfflogRL

In addition to all the advantages of Afflog, Afflog RL does not make use of Learning styles but uses RL to slowly adapt to the learning preferences of the user. Since it is very rare for a specific learning style model to exactly fit the learning preferences of an individual, an RL approach offers an alternative that, given enough training, may suit a user better. Another advantage of AfflogRL over its predecessor, is that once the State space is calculated, there is no need for planning or replanning since the structure of a course is given by the RL Policy, and thus there is no waiting time for the user. Finally, tutorials and tests in the domain model do not need to have any specific learning style properties

5.3 From Afflog to AfflogRL

AffLog RL is similar to AffLog, with the difference that AffLog RL replaces most ASP-driven control methods of AffLog with a Reinforcement Learning agent. In other words, the agent decides which tutorials and tests should be presented to the user on the spot. This implies that the system no longer has to generate a course with specific tutorials at the beginning using Planning and follow this course during its interaction with the user. Specifically, the planning, replanning and course selection programs are excluded in favor of a policy that the agent learns during interacting with users while utilizing a RL algorithm to learn said policy. Affective reactions are still employed, but will not happen automatically, but only if the agent chooses to perform them. In this work we do not present a new RL algorithm for RL Afflog. Instead, we focus on specific methods that can be utilized by different RL algorithms. These methods are:

- Generation of the state-space and action-space of the RL task. The *state space* is a *set* that contains all possible *states* that the tutor and the user can be in, excluding the emotional state of the user. This task is similar to the course planning task of Afflog in chapter 4, with the difference that the output is not just the completed sequences of tutorials and tests that a tutoring course consists of, but also all the incomplete sequences including those with wrong

tutorials and tests. However, unlike course planning, this task needs to be performed just once for a given set of training material of a subject. Action-space generation is the trivial task of adding all tutorials and tests available in the training material pool to the *Action-space* set, along with any available affective reactions.

- Limiting possible actions. RL algorithms usually have an action selection method that selects an appropriate action from the Action-space set given the current state of the agent. Instead of selecting an action from the whole set, we can take advantage of the logical structure of the domain and produce a subset that contains only actions that are viable to the current state and thus avoid selecting unnecessary actions.

- Updating the State of the world. This method is the same projection program used in Afflog. See 4.6.3).

- The reward function. As seen in chapter 3, the reward function is responsible for calculating a reward for an action that results in a new state of the world, thus reinforcing positively or negatively that action - state pair of the policy. In our case, the reward depends on two factors: The progress of the user, that is the number of successfully completed chapters by them, as well as their current emotional state. In other words, the closer the user is to finishing the course, and the more focused or happy they are, the bigger the reward.

5.3.1 Changing EC states and actions for the Reinforcement Learning task

The Action-space set contains the same as the actions used by afflog in chapter 4, except the affective responses. Specifically we have only kept three affective responses: “offer encouragement”, “offer praise” and the “terminate session” action. The “offer encouragement” action can be used in order to alleviate negative emotions such as frustration, boredom and confusion. “Offer praise” aims to prolong positive emotions such as flow and delight, and finally “terminate session” can be used if the user is experiencing frustration for an extended amount of time. We also added the “end Course” action that happens when the user completes the course so that the RL agent may know that the session has ended. Considering this, the number of actions available to the RL agent are $A = X+Y+4$, where X is the number of available tutorials of the course and Y is the number of available tests of the course.

As we discussed in chapter 4, a state is a set of Discrete Event Calculus fluents that describe the tutorials and tests that have been presented to the user thus far. These fluents also describe the comprehension of the user regarding said tutorials. Specifically, tutorials that belong to the same chapter as a correct test are marked as “correct”, tutorials that belong to the same chapter as a wrong test are marked as “wrong”, and tutorials still not validated by a tutorial are marked as “unknown”. Finally, tutorials marked as “wrong” will become “unknown” once revisited, since the system does not know if the user understood the tutorial this time. The only difference of Afflog’s states from AfflogRL’s states is that the former’s “currentEmotion” fluent does not appear in the latter's state space. This is done for two reasons: First, if we included the currentEmotion fluent to the RL state space we would have the current state space multiplied by 18, since we have 6 emotions with 3 levels of intensity for each emotion. Second, we want the emotional state of the user to be part of the reward function, and to be kept independent from the current state of the

world. However, since the currentEmotion fluent is needed for the reward function, and also for finding out which actions are available, we will differentiate between the RL current state s_{RL}, and the current state s so that:

$$s \in S, \quad s_{RL} \in S_{RL}, \quad S = S_{RL} \cup CE$$

Where, CE is a set that contains all the currentEmotion fluents. In theory, a state s_{RL} can be any combination of fluents of tutorials and tests. In practice however, these states are constrained from the structure of the course, and the consequences of the actions that caused them. For example, we cannot have a state that only contains two tutorials and none of those tutorials being the direct sequel of the other one. Also, these tutorials can only be marked as "unknown" since there are not any tests in the state to validate them.

5.3.2 State space

In order to calculate the state space we use a more generalized version of the planner that was used for creating the courses of afflog. That method essentially tried to find if there were tutoring courses completed within a certain timestep X, where at each timestep, the system could perform one tutorial or one test. By incrementing the timestep, eventually the planner would find all the possible correct courses from a given learning material pool.

The generalized method does not search for tutoring courses, but for states that can be parts of a tutoring course. For example, for X=0, the method will extract an empty state, as no tutorials or tests can happen, and put it in the state space set. This state will also be the starting state of the RL task. For X=1, the method will extract a number of *unknown* taught tutorials that begin at the first section of the first chapter, and put them in the state space set. For X=2, the method will extract sequences of tutorials and tests with a size of two that begin at the first section of the first chapter, and so on.

Also, whereas the Afflog planning method assumed that all tests were correct, the generalized method does not make that assumption. A user will inevitably fail a test at some point and this must be included at the state-space. This means that for each test, there are going to be two sequences. One where the test was correct, and all the tutorials that belong to the same chapter as this test are correct and one, where the same test and tutorials are wrong. For example let's consider the example in figure 5.1 We have two chapters Y and Z, four sections, sect 1 sect 2 and sect 3 in chapter Y and sect 4 in chapter Z. For timestep X=4 we have the following states:

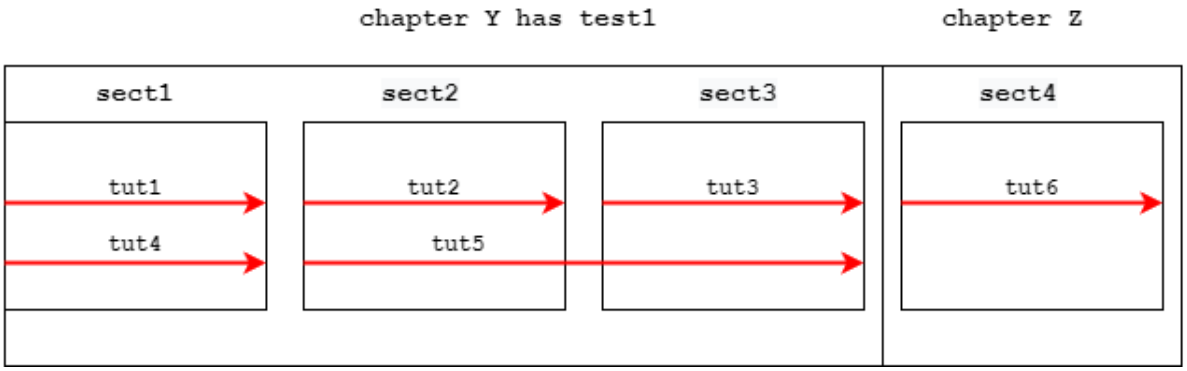

Figure 5.1: A simple course with 2 chapters, 4 sections, 6 tutorials, 1 test.

```
state A: holdsAt(taught(tut1,correct),4),
holdsAt(taught(tut2,correct),4),  holdsAt(taught(tut3,correct),4),
holdsAt(didTest(tst1,correct),4).

state B: holdsAt(taught(tut1,wrong),4),holdsAt(taught(tut2,wrong),4),
holdsAt(taught(tut3,wrong),4), holdsAt(didTest(tst1,wrong),4).

state C: holdsAt(taught(tut4,correct),4),
holdsAt(taught(tut2,correct),4),  holdsAt(taught(tut3,correct),4),
holdsAt(didTest(tst1,correct),4).

state D: holdsAt(taught(tut4,wrong),4),holdsAt(taught(tut2,wrong),4),
holdsAt(taught(tut3,wrong),4), holdsAt(didTest(tst1,wrong),4).

state E: holdsAt(taught(tut1,correct),4),
holdsAt(taught(tut5,correct),4),
holdsAt(didTest(tst1,correct),4),holdsAt(taught(tut6,unknown),4).

state F: holdsAt(taught(tut4,correct),4),
holdsAt(taught(tut5,correct),4),
holdsAt(didTest(tst1,correct),4),holdsAt(taught(tut6,unknown),4).

state G: holdsAt(taught(tut1,unknown),4), holdsAt(taught(tut5,wrong),4),
holdsAt(didTest(tst1,wrong),4).

state H: holdsAt(taught(tut4,unknown),4),holdsAt(taught(tut1,wrong),4),
holdsAt(taught(tut5,wrong),4),  holdsAt(didTest(tst1,wrong),4).

state I: holdsAt(taught(tut4,unknown),4), holdsAt(taught(tut5,wrong),4),
holdsAt(didTest(tst1,wrong),4)

state J: holdsAt(taught(tut1,unknown),4),holdsAt(taught(tut4,wrong),4),
holdsAt(taught(tut5,wrong),4),  holdsAt(didTest(tst1,wrong),4)
```

States **A** and **B** consist of the same tutorials, In state **A** test **tst1** was correct and so all the taught tutorials of chapter Y are correct. In contrast, in state **B**, **tst1** was wrong, and so all the

same tutorials are wrong. Such is also the case for states **C,D** with **tut4** replacing **tut1**. States **E** and **F** reach chapter Z after completing **tst1.**Finally, states **G, H, I, J** show what is the next action after failing a test. States **H** and **J** try tutorials that have not been taught before, while states **G** and **I** retry the same tutorial, and as a result, these states include three predicates and not four with said tutorial being unknown and no longer wrong.

5.3.3 Limiting possible actions for action selection, and updating the state of the world.

Once the state space is known, the system can form a RL policy and start updating it in sessions with users. Each session starts from state 0, which is an empty state because no tutorials have been taught, or another state in case of a returning user that has started not yet finished the course. In both cases, the next step is to use another ASP program to find the *current action space,* which is a subset of the action-space set that contains all the legal actions taken from the current state. We will call this ASP program *action limiter*. In this case, the current state is "state 0" and the resulting current action space set will be all the tutorials that start at the first section of the first chapter.

It is important to note that by limiting the current action space, the action limiter ensures that the structure of the tutoring course will be kept stable, by ensuring that the sequence of chapters, sections, tutorials and tests remains correct while keeping the tutoring course as flexible as possible. In case of a test done wrong, the action limiter will also deviate from the taught tutorials and tests by selecting tutorials and tests that have not been yet taught to the user if such tutorials and tests are available. The action limiter will also let emotional reactions in the current action space according to the current emotional state of the user.

After the current action space is found ,then the chosen RL Action selection method selects an action from that subset, and the tutor performs the selected action (see chapter 3 for different action selection methods used in RL). If the action is a tutorial or test, the tutor presents it to the user through the GUI, and if it's an emotional action, the tutor offers praise or encouragement to the user according to the user's current emotional state.

After the action is performed, the agent will update the state of the world by using the projection program from chapter 4 by taking the current action as input and getting the new current state as output.

5.3.4 Reward Function

The reward function is vital to a Reinforcement Learning agent, as it dictates which actions are more desirable for the current state of the agent in order to eventually reach its desired goal. Reinforcement Learning agents that are used as tutors, often use, among other metrics, user progress, time spent in session, learning gains, correct and incorrect answers and others in order to calculate the reward signal ,[79]. However, since Afflog RL is an affective tutor, we have decided to make use of the current emotional state of the user by incorporating it into the reward function. This is not something new as many works in affective computing, and human-robot interaction are driven by the user's emotions. Therefore, Afflog RL's reward function makes use of both the user's progress and the user's current emotional state. Specifically, we track the user's progress as a percentage of the completed course and call it $learning$. while the current emotion of the user is

called $engagement$. Both of these metrics are the same and their sum is the final reward signal r as seen in equation (1):

$$r = 0.5 * engagement + 0.5 * learning, \quad -50 \leq r \leq 100 \qquad (1)$$
$$-100 \leq engagement \leq 100, \quad 0 \leq learning \leq 100$$

$learning$ can only have a zero or positive value and it depends on the number of correct tests and the number of tutorials associated with these tests as seen on equation (2):

$$learning = correctAssociatedTutorials/totalTutorials * 100. \qquad (2)$$

Where $correctAssociatedTutorials$ is the sum of the tutorials that belong to the chapters of the correct tests, and $totalTutorials$ is the sum of all tutorials of the learning material pool. This way, if no correct tests are done, the number of associated tutorials will be zero, and thus $learning$=0, while, for each correct test done, $learning$ will be equal to a higher percentage of tutorials. If the user finishes the course, then every test will be correct and thus $correctAssociatedTutorials = totalTutorials$. The reason why the associated Tutorials metric was used is because not all chapters have the same size or difficulty. The more tutorials a chapter has, the more important it should be.

We also make use of this formula with the knowledge that all chapters should have a proportionate number of tutorials. That is, the number of different connected tutorial sequences for each chapter should be roughly the same. If for some reason a chapter has many alternative tutorials and the others have only one sequence of connected tutorials and no alternatives, then, this chapter will be overrepresented. In this case, the course creator should add more tutorials to the underrepresented chapters, or find another way to calculate the importance of each chapter.

$engagement$ can either have a positive or negative value depending on the current emotion of the user and its persistence (or valence). In chapter 4, we saw that the affective reactions of Afflog to the current emotion of the user, varied according to how much that emotion stayed the same from one tutorial to the next. Specifically the currentEmotion fluent could have a persistence level of low, medium or high. To calculate the engagement level of the user, we quantified the persistence levels as well as the current emotion. And used equation (3).

$$engagement = emotion * persistence \pm 0.1; \qquad (3)$$

Where persistence can have the value of ‘1’ if low, ‘2’ if medium and ‘3’ if high, while the emotions are quantified as: Neutral = 0, flow = 33.3, delight = 15, bored = -20, confused = -10 and frustrated = -33.3.

The reasoning behind these values is that the system aims to bring the emotional state of the user closer to the ‘flow’ state, which according to [11], is the state better suited for learning. On the other hand, the ‘frustrated’ state is the least suited emotional state for learning and thus has the lowest quantified value. ‘Delight’ has a positive value, although not as high as ‘flow’, and the neutral state is quantified as zero. Regarding the ‘bored’ and ‘confused’ states, both have a negative impact on

learning , [31], and thus both have negative values. The ‘confused’ state has a slightly higher value than ‘bored’ because it has been reported that it is probable for confused learners to redouble their efforts in order to better understand the lesson, and enter the state of ‘flow’. These values guarantee an engagement score between -100 and 100. However, If the user reaches the end of the course, the final reward signal will be the maximum regardless of the emotional state of the user.

5.4 Afflog RL functionality and architecture

5.4.1 Functionality

As seen in 5.2, Afflog RL consists of offline tasks that are performed once in order to initialize the RL agent, before user interaction, and online tasks that are performed by the agent for each user session. Remember that these tasks concern only one course subject. A different subject should start the whole process from the start. In this workflow we assume that we use Q- Learning, or another TD- learning method.

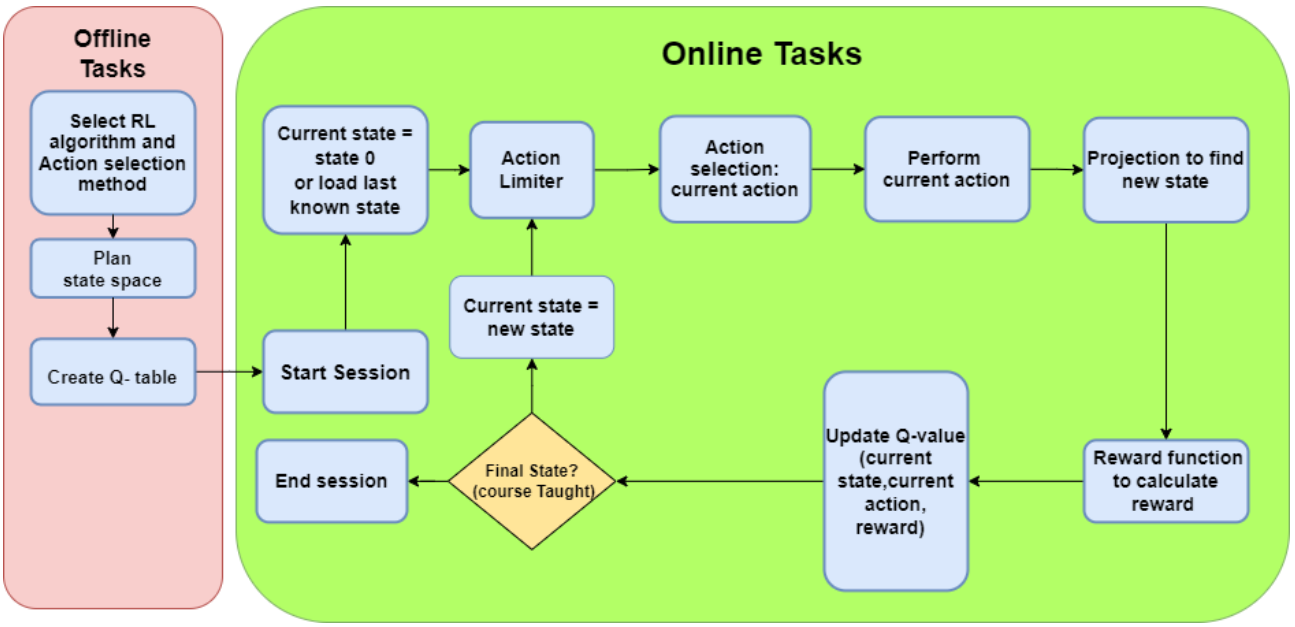

Figure 5.2 : Workflow of the offline and online tasks of Afflog RL.

Offline tasks include selecting the appropriate RL algorithm, selecting the appropriate action selection method, as well as the different algorithm variables such as the learning rate, the discount factor, and the initial values of the Q-table. After this, the methods described in 5.2.1 and 5.2.2 are

utilized in order to create the action space and plan the state space. Finally, the Q-table is created from which the policy of the RL agent can be derived.

Online tasks are all the tasks of the RL agent done within a session with a user. At the start of the session, if it is the first time that the user sees this course the current state of the agent is set to state 0 which is the empty state. If the user is returning to the course, the agent will load the last known state. After the current state is known, the agent will use the *action limiter* in order to find all the available actions from that state and update the *current action space* set. This set is then used as input for the action selection method, which selects the current action from the set.

The agent then performs the action by using the GUI to interact with the user. The interaction will either be a tutorial, a test, or an emotional response. After the interaction takes place, the user will either request the next tutorial or test. The agent then will display the test's results to the user if a test was performed, and then will proceed to use the projection program in order to find the new state that this action will lead to.

After this, the reward function will calculate the reward of the current action, based on the new state, and the agent will use this reward to update the Q-table. If at this point the course is finished, the agent will end the session, otherwise the new state will become the new current state and the process will continue until the session ends either by finishing the course, or the user terminating it.

5.4.2 Architecture

Figure 5.3 depicts AffLog RL from an architectural point of view. The System's modules are written in Java, while the Knowledge Representation parts, including the domain model as well as various methods of the tutor module, are written as ASP logic programs using the Clingo System to solve them.

The User Interface (UI) layer consists of Afflog RL's Interface. The interface is similar to that of the Afflog system. The differences are that the file directory containing the tutorials and tests of the current course starts empty, but fills with each tutorial and test presented to the user. These tutorials and tests will be available to the user for revision. Also there is no need for a "create course" or "replan" button but neither a progress bar, since it is impossible to predict the exact size of the resulting course.

The tutor layer contains all the methods and data structures that are responsible for controlling the system, including the RL agent. The tutor layer also contains methods that call the answer set programs through the parsing layer. Among other data, it contains the Q-table of the RL agent as well as the current action, and the current state. From a traditional ATS architecture perspective, the tutor layer contains the tutor model, the student model and the affective model of the ATS.

The parsing layer contains all the methods that connect the tutor layer with the Answer Set programs. These methods create and write the input files of the Answer Set programs according to the tutor layer but are also responsible for receiving the output of the Answer Set programs and transforming it to java data types, such as int, String, etc.

Afflog RL makes use of four Answer Set Programs. Each of these programs consists of a number of files with the .lp extension. Some of these files such as the domainModel.lp and DEC.lp files are used in more than one of these programs. These are the State space planner, the course limiter and the projection described in 5.2, and 5.4. The last ASP program, the testHasAssociatedTutorials is used in order to calculate the number of associated tutorials for each test used by the reward function to calculate the *learning* variable.

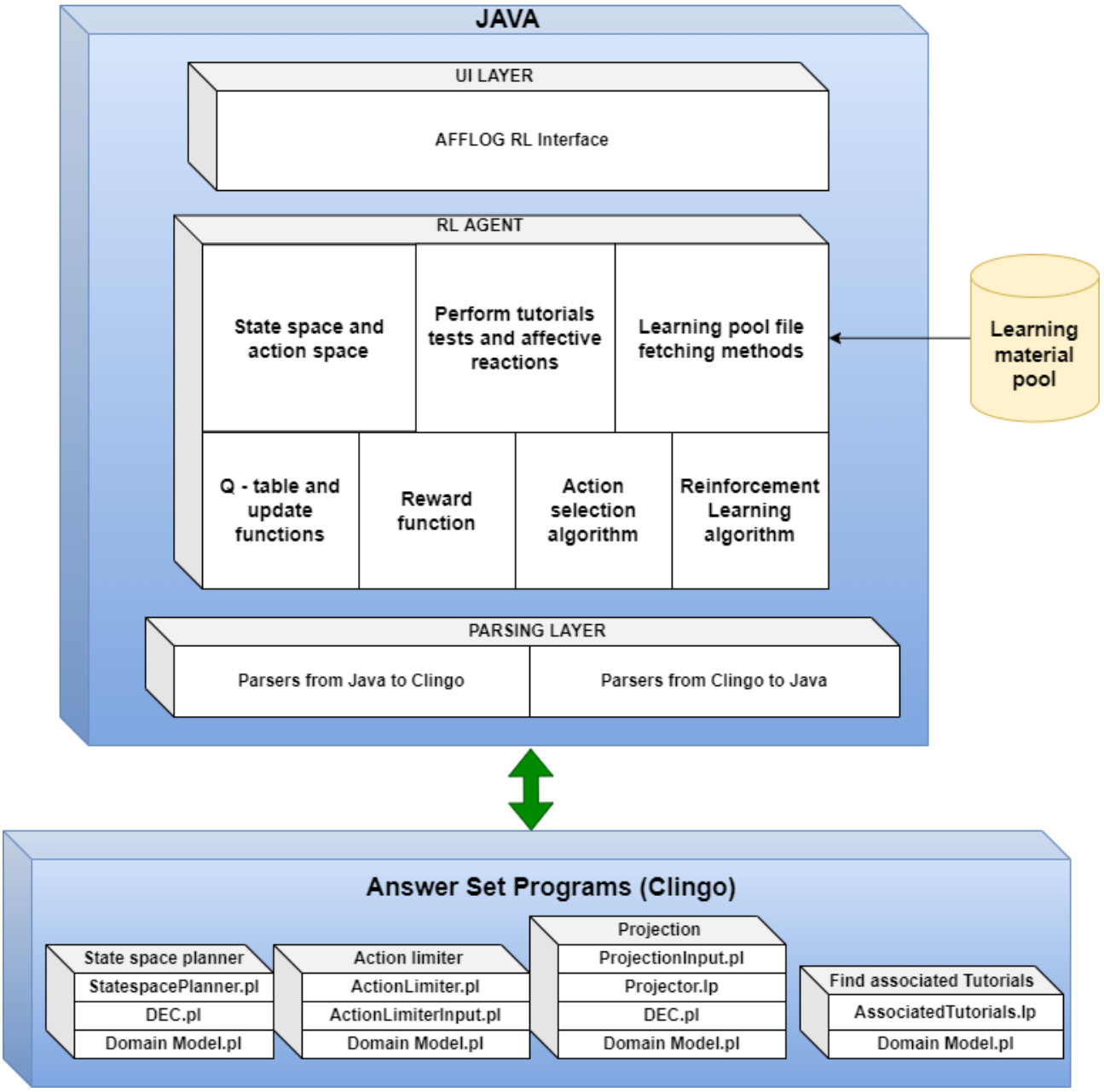

Figure 5.3: Afflog RL Architecture.

5.5 Methods and ASP programs

In this section we elaborate on the three Answer Set programs in order to give to the reader a measure of understanding of their usage. We will not cover the projection program as it is the same as the one addressed in section 4.6.3. Again the Domain model is utilized by all ASP programs, while only the state space planner uses the Discrete Event Calculus.

5.5.1 Associated Tutorials

The **testHasAssociatedTutorials** program finds the number of tutorials that belong to the same chapter as a test. This number is used by the reward function in order to calculate how much of the course the user has understood.

The program consists of 2 rules. The first rule **(1)**, creates all the possible **testAssociatedwithTutorial/2** predicates, where each predicate contains a test and a tutorial that belong to the same chapter. The second rule **(2)**, uses clingo's soft constraint **#count** in order to sum all the tutorials that appear at a **testAssociatedwithTutorial/2** predicate with the same test. The output of the second rule, and of the program, is the **testHasAssociatedTutorials/2** predicates, instantiated a number of times as the number of tests available at the domain model. In our setup, using the catan domain model, the program takes 24 ms to run on average.

```
testAssociatedwithTutorial(TEST,TUT):-
test(TEST);hasChapterTest(CH,TEST);
hasChapterSection(CH,SECT);hasTutorialStart(TUT,SECT);section(SECT);
chapter(CH);tutorial(TUT).                                          (1)

testHasAssociatedTutorials(TEST,TUTORIALS):-TUTORIALS=
#count{TUT:testAssociatedwithTutorial(TEST,TUT)},test(TEST).        2)
```

5.5.2 State Space Planner

State space planner is identical to the planner described in 4.6.2 with the following modifications:

1. Unlike the previous planner, the State space planner does not use constraint rules to stop wrong tests from happening.
2. *Initiates* and *terminates* DEC rules must be written for wrong Tests. Specifically, a rule stating that if a test is wrong, then all unknown taught tutorials that are part of this chapter are terminated. Also, Another similar rule must be written, that initiates the same previously unknown tutorials as wrong tutorials.

   ```
   terminates(doTest(TST,wrong),taught(TUT,unknown),T):-
   holdsAt(taught(TUT,unknown),T);
   ```

```
hasChapterTest(CH,TST);
hasTutorialStart(TUT,SECTSTART);
hasTutorialEnd(TUT,SECTEND);
hasChapterSection(CH1,SECTSTART);
hasChapterSection(CH2,SECTEND);
1{CH=CH1;
CH=CH2;
containsChapter(TUT,CH)};
time(T).

initiates(doTest(TST,wrong),taught(TUT,wrong),T):-
holdsAt(taught(TUT,unknown),T);
hasChapterTest(CH,TST);
hasTutorialStart(TUT,SECTSTART);
hasTutorialEnd(TUT,SECTEND);
hasChapterSection(CH1,SECTSTART);
hasChapterSection(CH2,SECTEND);
1{CH=CH1;
  CH=CH2;
containsChapter(TUT,CH)};
time(T).
```

3. A constraint rule that prevents a test that was wrong once to be wrong again, in order for the planner not to be stuck in a deadlock (a never ending loop).

```
:-happens(doTest(TST,wrong),T);holdsAt(didTest(TST,wrong),T).          (3)
```

5.5.3 Course limiter

Course limiter is an ASP program which, given the agent's current state, will find all possible legal actions that the agent can take. The program does not use the Discrete event calculus because it does not have to find a sequence of multiple events, but only the next possible action.

A typical program input includes the tests and tutorials done, as well as the current emotional state of the user (4).

```
doneTestWrongly(test07).
doneTestCorrectly(test01).
doneTestCorrectly(test03).                                    (4)
doneTestCorrectly(test06).
inputTutorial(tut21).
inputTutorial(tut22).
currentEmotion(flow,low)
```

First, the program tries to understand if a tutorial needs to take place. If there are no input tutorials available (rule 5), this means that we are at the empty state (state(0), and the program will just

output all the tutorials that start at the first section in the first chapter. In this case, the first section in the domain model file is named **start.**

```
doTutorial(TUT):- not inputTutorial(_);                    (5)
                  tutorial(TUT);
                  hasTutorialStart(TUT,start).
```

Rule 6, determines if the agent can reach the end of the course. In this case, the program will only return a single action **end**, which signifies the end of the course. In order for this rule to fire, the last test of the last chapter must be correct.

```
end(0):-doneTestCorrectly(TST);                            (6)
        hasChapterTest(CH,TST);
        chapterEnds(CH,end).
```

In this program, tests take precedence from tutorials. Rule **(7)** states that if a test cannot be done, and if the end of the course cannot be reached, then a tutorial that starts where an input tutorial ends can be done.

```
doTutorial(TUT):- not doTest(_);                           (7)
                  inputTutorial(TUT0);
                  hasTutorialEnd(TUT0,SECT);
                  tutorial(TUT);
                  hasTutorialStart(TUT,SECT);
                  not inputTutorial(TUT);
                  not end(0).
```

The following rules ensure that no test deadlocks will happen. Deadlocks happen when all the tests of a chapter are wrong. In that case, both tests are eligible to be performed again. All rules after **(7)** and before **(8)** create helper predicates that can be used in order to create the two deadlock rules **(8)** and **(9)**. Specifically the **deadlock1** predicate is activated if at least two tests of the same chapter are wrong. **deadlock2** is activated when there is only a single test in that chapter, and this test is wrong. Finally, Rule **(10)** lets a deadlocked test happen.

```
doneTest(TST):-doneTestCorrectly(TST).
doneTest(TST):-doneTestWrongly(TST).

moreThanOneTests(TST):-hasChapterTest(CH,TST);
                       hasChapterTest(CH,TST1);
                       TST1!=TST.

%There are more than one wrong tests for this chapter.
samechapterWrongTestsExist (TST,TST1):-  hasChapterTest(CH,TST);
                                         hasChapterTest(CH,TST1);
                                         test(TST);
                                         test(TST1);
                                         TST1!=TST;
```

```
                                        doneTestWrongly(TST1).

deadlock1(TST):- doneTestWrongly(TST);                    (8)
                 test(TST1);
                 samechapterWrongTestsExist (TST,TST1).

deadlock2(TST):- doneTestWrongly(TST);                    (9)
                 not moreThanOneTests(TST).

doTest(TST):-1{deadlock1(TST);                           (10)
               deadlock2(TST)};
                   test(TST).
```

The following rules determine when a test can happen. Again, we have helper predicate rules such as **similarTestDone,previousTestNOTDone,chapterBefore,** and **transchapterBefore.** However, the three main rules are the three **doTest** rules (11-13). Each one of those rules determines if a test should be performed:

1. Rule **11** states that a test should be done if a tutorial ends at the end of a chapter, and no other tests of this chapter have been done.
2. Rule **12** states that a test that belongs to chapter 'A' should be done, if a tutorial begins at chapter 'A', and ends at some other chapter, and no other tests of chapter 'A' have been done.
3. Rule **13** states that a test that belongs to chapter 'B' should be done, if a tutorial begins at chapter 'A' that is before 'B', and the ame tutorial ends at chapter 'C' that is before chapter 'B'.

```
similarTestDone(TST1):-doneTestCorrectly(TST);
                       hasChapterTest(CH,TST);
                       hasChapterTest(CH,TST1);
                       TST!=TST1.

previousTestNOTDone(TST2):-not similarTestDone(TST1);
                           not doneTestCorrectly(TST1);
                           hasChapterTest(CH1,TST1);
                           hasChapterTest(CH2,TST2);
                           chapterBefore(CH1,CH2).

chapterBefore(ch1,ch2).
chapterBefore(ch2,ch3).
chapterBefore(ch3,ch4).

%case: The input tutorial finishes at the end of a chapter. A test is
needed.
doTest(TST):-inputTutorial(TUT);
(11)
             hasTutorialEnd(TUT,SECT);
             chapterEnds(CH,SECT);
```

```
            hasChapterTest(CH,TST);
            not doneTest(TST);
            not similarTestDone(TST);
            not previousTestNOTDone(TST);
            not end(0).

%case: The input tutorial may overlap a chapter. A test is needed for the
tutorial's origin chapter.
doTest(TST):-inputTutorial(TUT);
(12)
            hasTutorialStart(TUT,SECT0);
            hasTutorialEnd(TUT,SECT);
            hasChapterSection(CH,SECT);
                 hasChapterSection(CH0,SECT0);
                 CH0!=CH;
                 hasChapterTest(CH0,TST);
                 not doneTest(TST);
                 not similarTestDone(TST);
                 not previousTestNOTDone(TST);
                 not end(0).


%case:The input tutorial may overlap a chapter. A test is needed for a
chapter between 2 others.

transchapterBefore(CH1,CH2):-chapterBegins(CH2,SECT);chapterEnds(CH1,SECT
).
transchapterBefore(CH1,CH3):-transchapterBefore(CH1,CH2);transchapterBefo
re(CH2,CH3).
transchapterBefore(CH1,CH4):-transchapterBefore(CH1,CH2);transchapterBefo
re(CH2,CH3);transchapterBefore(CH3,CH4).

doTest(TST):-inputTutorial(TUT);
(13)
            hasTutorialStart(TUT,SECT0);
            hasTutorialEnd(TUT,SECT);
            hasChapterSection(CH1,SECT);
                 hasChapterSection(CH0,SECT0);
                 transchapterBefore(CHX,CH1);
                 transchapterBefore(CH0,CHX);
                 hasChapterTest(CHX,TST);
                 not doneTest(TST);
                 not similarTestDone(TST);
                 not previousTestNOTDone(TST);
                 not end(0).
```

The following rules dictate whether an affective reaction should take place. Providing that the agent has not reached the goal state.

```
doRespond(encouragement):-1{currentEmotion(confused,low);
                            currentEmotion(frustrated,low);
                           currentEmotion(bored,low)};
                           not end(0).

doRespond(praise):-1{currentEmotion(flow,low);
                     currentEmotion(delight,low)};
                     not end(0).

doRespond(terminateSession):-1{currentEmotion(confused,high);
                               currentEmotion(frustrated,high)};
                               not end(0).
```

5.6 Limitations

Some limitations of AfflogRL are similar to the limitations of its predecessor. Specifically, courses need to have multiple tutorials covering each subsection in order for the RL agent to choose the tutorial best suited to the needs of the user. Also, again the affective part of the system is not connected to any affect-sensing mechanism but it relies on the user or an expert monitoring the user.

The main limitation of the system is that it is not currently optimized in order to quickly update the Q-values of the Qtable, and completely relies on the chosen RL algorithm for value update. This would not be a problem if training could be done offline, for example, with simulated users, but since Afflog is using interactions with real users, it is hard to accumulate many episodes, and thus hard for the algorithm to converge to an optimal solution. Moreover, as the number of tutorials and tests in a course increase, the state space also increases, making convergence even harder.

Another problem is that even if Afflog RL is trained adequately in a single course for a specific user, it will not retain any information that could be used in other courses with the same user, or to other users with the same course.

Chapter 6
Evaluation of the Afflog and AffLog RL Systems

In this chapter, we evaluate the two systems presented in this dissertation. First we state the *purpose* of the evaluation, its *scope*, and what are our expectations regarding the results. Second, we present the course that was created as a use-case for the evaluations. Third, we state the preparations for the evaluations. In the fourth and fifth sections we explain the evaluations of Afflog, and Afflog RL, respectfully. Finally, in the sixth session we present and explain the results of the evaluation.

6.1 Purpose (Motivation)

This evaluation has multiple purposes. First we would like to evaluate the Afflog system, to determine if the users will be able to learn using it while also having an overall positive experience. To determine the quality of learning we will measure the learning gains [8] of 20 users after their interaction with the system during a forty minute session. Learning gains is a measure used frequently by e-learning and ITS systems, [8], and is the difference of the user's knowledge of the subject after the learning session minus their knowledge before the session. Learning gains can be summarized as the improvement in knowledge, skills, work-readiness and personal development made by students. To measure user experience we will use a UX questionnaire after the session, [68]. Also, for the purpose of this evaluation, we created a course that explains the rules of the Settlers of Catan, [66], board game. We expect a medium to high learning gain for each user, as well as an overall positive user experience.

The second part of the evaluation is to measure the user-experience and learning gains of the Afflog RL system. To do this we first have to train the system with real users, and then perform an evaluation similar to that of Afflog's. In order to train the system, we are going to perform 100 40 minute sessions with users, and 20 more sessions to evaluate the trained system. To avoid overfitting, a single user cannot interact with the system for more than one session, so in total we will have 140 different users to evaluate both systems. We will then measure the scores of the two systems and find which system performs better under those circumstances.

Bear in mind that this is not the optimal way of using Afflog RL, since it was designed to be trained by and eventually adapt to a single user. For that to happen it would require a large number of sessions (more than a hundred) per user with a number of different courses more complex and

greater in size than the available use-case course. In our case, we trained Afflog RL to teach a single course to a hundred different users. What we expect as a result is an Affective tutoring system that chooses the tutorials, actions and affective reactions that will better suit the majority of the user base, and not to adapt to a group of users. Regarding learning gains we expect roughly equal or greater learning gains than the Afflog System, and a User satisfaction score similar to that of Afflog.

Another thing to consider is that the Afflog RL evaluation is done primarily to test the RL agent and its different components with no optimizations. During evaluations we used a single RL algorithm, a single action-selection method, and only one set of RL variables such as the learning rate, discount factor and initial conditions. In order to optimize Afflog RL, different algorithms, methods and variables should be used as well as techniques to generalize the Q-function such as Relational reinforcement learning (RRL), [7] .

6.2 Use-case course

To evaluate the two systems a use-case course in English on how to play Settlers of Catan was created. Settlers of Catan, [66], is a German board game which, while simple to learn, is quite difficult to master as players have come up with a variety of strategies in order to win. It also represents a task not usually tackled by e-learning systems, although the idea that such systems should expand beyond teaching courses into other areas such as assembly instructions, fitness coaching and entertainment is gaining ground.

The methodology for creating the course was to first divide it into chapters, then find the smallest possible parts of these chapters in order to define the sections, and then to find or create tutorials of different modalities, and empirically determine where each tutorial starts and where it finishes. A group of players were consulted, and asked to rate each tutorial on a scale of 1 to 3 according to its duration and difficulty. Finally a number of multiple choice tests were created for each chapter. Although more complex tests could be supported such as exercises, playing sessions, etc., as their state in the domain model remains the same, we wanted to keep the users as calm and relaxed as possible

We divided the course into 4 parts. The first part explains how to set up the various pieces of the game, the second is about starting the game, the third part describes all the rules, and the fourth part presents a number of strategies. In order to complete the course, the user must successfully complete a chain of tutorials starting at the first chapter and finishing at the end of the third chapter. The tutorials that continue to the fourth chapter are optional and their respective courses will only be used if the user decides to do so when prompted by the system. In total, 47 tutorials were created, and 7 multiple choice tests.

Fortunately there is a great number of multimodal online tutorials for Settlers of Catan. Special mention must be given to Wikihow [13] that provided most of the material for the creation of this course, as well as channels on youtube.

6.3 Before The evaluation

6.3.1 Usability testing

To evaluate both systems, we first had a group of three experts from FORTH's Human Computer Interaction Laboratory perform a Heuristic evaluation, [80], of both of the systems in order to identify usability problems of the GUI. This is an iterative process in which a small number of evaluators examine the interface and judge its compliance with recognized usability principles. Once the usability problems are identified, the GUI is redesigned according to their suggestions, and then the process is repeated. For more information on Heuristic Evaluation please refer to [80, 81]. Overall, 26 problems were identified and were rectified.

6.3.2 Pre-test , post-test and UX questionnaire

In order to compute learning gain, we prepared two identical online questionnaires using google forms with 10 multiple choice questions on the Settlers of Catan to be used as post test and pre-test. Six of those questions also appear as multiple choice tests for the Settlers of Catan course.

A user experience (UX) questionnaire collects quantitative and qualitative data about a user's interactions and experience with a website or digital product. UX survey data supports and complements website analytics and UX metrics collected through methods like A/B testing, and session recordings [68] . For the purposes of that evaluation we created a simple UX questionnaire of five questions with multiple choice answers, in order to measure the overall experience of the users. Copies of the pre-test, post-test and UX forms can be found on the Appendix of this dissertation.

6.3.3 Consent forms

After consulting with the Research ethics committee of the University of Crete, [82], we created two consent forms, one for the data collection to be used for the training of Afflog RL, and one for the evaluation of both systems. The consent forms describe the nature of this research, the tasks required of the users, how their personal data as well as the data generated from their interaction with the systems is going to be used and stored, as well as more information such as how to contact the researchers. Copies of both the consent forms can be found on the Appendix of this dissertation.

6.3.4 Setup

In order to perform the evaluations, we ran the ATS on a laptop with an AMD Ryzen 5 3550H and 16 GB RAM, using Netbeans 8.0.2 and Clingo 4.5.4 for Windows. We used the preferred Voice Over Ip application of each user such as Skype, facebook messenger and discord, [87, 89, 90], to talk to the volunteers during sessions, and the Anydesk, [88], remote desktop application in order to allow the volunteers to interact with the ATS. Anydesk was chosen because of its simplicity and usability. The Settlers of Catan course has 58 actions 47 tutorial actions, 7 test actions, 3 affective reactions, and 1 action that signifies the end of the course. Also when the "Plan state space" ASP program ran, 697 possible states were found as the state space.

It is important to note that the runtime of each of the ASP programs using Clingo, was less than 1 second, and thus the transition from one tutorial to the next was done seamlessly to the users. The only exception to this, are the Planning ASP programs. Considering the use-case course, the Afflog

Planner can take up to 8 seconds to plan all available courses according to the learning style of the user, however this planning can take place before the user session as long as the user's learning style is identified, and the learning material pool of the course (and thus the domain file) has not changed. Still, the replanning needed to create new courses after a wrong test, will cause the user to wait for up to 8 seconds. As seen in chapter 5, Afflog RL only uses the planner to calculate the state space of the RL tutor. This can be a time demanding task considering the other ASP reasoning tasks as it took over 2 minutes to calculate the state-space of the use-case course. However, this is done only once for the same learning material pool, and will happen during a session.

6.3.5 Afflog RL training

In order to train the Afflog RL Tutor, we collected data from real users. Specifically, for the data collection, 101 volunteers participated in online sessions from their computers or mobile phones using the AnyDesk program [88] where they interacted with Afflog RL as users. Only 9 of the users had played the game before and only two stated that they remembered the rules of the game well. Each session lasted for up to one hour, where the first 15 minutes were used to familiarize the user with the task and the GUI, and the rest 45 minutes used to interact with the Tutor. During the interaction, the user had to follow the tutorials presented by the tutor in order to complete the course. An expert was also present during the session in order to offer guidance to the user regarding the GUI if the user was stuck. The expert however did not offer any help regarding the contents of the course.

For training AfflogRL's policy we used the AfflogRL agent from section 5. The Q-Learning Reinforcement Learning algorithm was used as presented in [83] and implemented by Chen in [84], with a constant Learning rate α=0.1, a discount factor γ =0.7 and initial conditions Q_0=0.1. For action selection, we used the Softmax method using a Gibbs distribution. As stated in 6.1, we used Q - learning and softmax algorithms with these variables as a way to test the Afflog RL learning and thus they may not be the optimal tools for such a task. Q - learning was used as it is one of the most widely used RL algorithms. As for the variables, we used a typical (as described in the work in, [83]) constant learning rate of α=0.1, a discount factor γ =0.7 that will make the algorithm strive for a long term high reward rather than the short term reward. The softmax action selection algorithm was used because it is better than a greedy approach in the sense that although the greedy action (the action with the highest Q-value) is still given the highest selection probability, the other actions are ranked and weighted according to their value estimates. An e-greedy action selection would rank the other actions equally, and thus could lead to some bad actions being taken.

6.3.6 User Information

All users were adult men and women of ages 20 to 72 who volunteered to take part in the training and evaluation of the systems online. All of the users were colleagues or family of the author. User sex was 55% male and 45% female, while user age followed a normal distribution. 99% of the users were Greek and although they did not speak English as a primary language, the vast majority of them had no problem understanding the course. If a user had difficulty understanding a word or a phrase of the course due to not being familiar with the language, the course expert would translate said part in Greek. Over 90% of the users did not have any prior experience playing Settlers of Catan, and most of them that had played the game before, could not remember all the rules correctly.

6.4 Afflog Evaluation

In order to evaluate the Afflog system, 21 volunteers participated in online one-hour sessions where they used the Anydesk program to interact with the Afflog System. After their interaction, each user was given a post test questionnaire in order to measure their learning gains.

Before the online-session, each volunteer was given ample time to understand the task that they had to perform, as well as read and sign the consent form further explaining the task. Also, each student had to complete Felder and Soloman's 'Index of Learning Styles Questionnaire' [1], found online at "https://www.webtools.ncsu.edu/learningstyles/" and mail the results to the author. The author then quantified the results into ASP predicates using the user profile form (section 4.4) in order to create the student model of the user as seen in section 4.2.

For the online-session, after establishing a voice-over Ip call and using anydesk to connect to Afflog, the user was again informed of the task, and was given the pre-test. If a user did not have any prior experience with Settlers of Catan or similar games, the pretest score was set to 0. After the pretest, the session was similar to that of a training session in 6.3.6. The user spent some time understanding the UI, and then was left to interact with the system. Again, an expert was available to offer guidance regarding the UI and the Tutor. The only difference from the training sessions regarding the interaction with the system was that the user had to first create and select a course that matched their learning style. For more information on starting to use the Afflog system, see 4.7.1. The user was given approximately 20 minutes to do the pre-test and understand the UI of the system, and up to 40 minutes to interact with the system. From the 21 users, only 3 did not manage to finish the course in the given time and 2 of these users had connection problems and had to restart their devices. Regardless, after the 40 minute mark, users had to stop using the system, and fill the post-test form.

After their interaction with the system, the users were given 5 minutes to fill the post-test and the UX questionnaire form, increasing the maximum time of a session to 65 minutes. A user spent on average 15 minutes for the pretest and understanding the system, 36 minutes for interacting with the system, and 3 minutes to finish the post-test and the UX form, making the average session 54 minutes. Of the 21 volunteers, only 6 have played Settlers of Catan before and none managed to score perfectly on the pre-test. All the questions of the pretest and posttest were not randomized.

6.5 Afflog RL Evaluation

In order to evaluate the Afflog RL system, 21 volunteers participated in online one-hour sessions where they used the Anydesk program to interact with Afflog. After their interaction, each user was given a post test questionnaire in order to measure their learning gains. The evaluation was similar to that of Afflog (6.4) with the difference that the volunteers did not have to complete the learning style questionnaire, and also did not have to choose a course. Again the maximum time of a session was 65 minutes. 20 for pre-test and familiarizing with the UI, 40 minutes interaction with the system, and 5 minutes for the post-test.

The average time that each user spent for each session was similar to that of the Afflog evaluations, with the difference that the average time spent using the system was reduced by 2 minutes down to 33 minutes. This is probably due to the Afflog RL UI being simpler than that of Afflog, due to it not having a progress bar and the plan course button, and also not performing so many affective reactions, or replanning the course when a test was wrong. Of the 21 volunteers, only 3 have played Settlers of Catan before and none of those managed to score perfectly on the pre-test. All the questions of the pretest and posttest were not randomized.

6.6 Results and discussion

6.6.1 Results

After all data was collected we computed the normalized gain of averages as described by Hake in [86]. Specifically, to calculate the Gain of averages, first we calculated the average pre-test and average post-test score for all of the users of a system, and then take the normalized gain ⟨g⟩. The latter is defined as the ratio of the actual average gain (%⟨post⟩−%⟨pre⟩) to the maximum possible average gain (100−%⟨pre⟩). Here, brackets indicate class averages.

An alternative calculation to the normalized gain of averages, as presented by Bao, [85], is to calculate the normalized learning gain for each user and then take the average:

$$g_{ave} = \langle(\text{Post} - \text{Pre})/(100 - \text{Pre})\rangle$$

The difference between these two calculations is not significant for a high number of users, but may differ quite a bit for a small number. According to, [85, 86], a normalized learning gain value of <.3 is considered small, a value of 0.3 to 0.6 is considered medium, and a value > =0.7 is considered large. The Learning Gain values of the evaluations are displayed in Table 1. The Results of the Ux questionnaire are displayed in table 2.

	Afflog	*Afflog RL*
Average of gains	0.7667	0.7772
Gain of averages	0.7816	0.7861

Table 6.1 : Learning Gains.

We performed a one tailed and two tailed Unmatched T- Test between the Post - pre scores of Afflog and AfflogRL with the null hypothesis being that there is no significant difference between the two scores. Indeed Both test results found that there is no significant difference between the two scores with P > 0.05. Specifically in the one tailed t-test the t-value is -1.05316. and the p-value is 0.149458 while in the two tailed t-test the t-value is -1.05316 and the p-value is 0.298917.

The user experience questionnaires showed that the majority of the users (over 60%) found the course interesting, exciting, and enjoyable. There were no major differences in user experience between the two systems except a slight difference in the excitement levels, with Afflog RL scoring more. This could be attributed to the large number of affective reactions and planning time of the Afflog system.

6.6.2 Discussion

The results of the evaluation show that both systems had high learning gains. But since there is not a significant difference between the learning gains of both systems, and their gain values are close, we cannot claim that one system is better than the other.

Considering the small size of the training set, the small number of tutorials available in the Settlers of Catan course, and the lack of adaptation to the specific user, it appears that AfflogRL has the potential to perform better than the Afflog system if these issues can be rectified. However, in order to verify this, more experiments must be performed for both systems in order to ensure more conclusive results.

With only 100 training sessions, the RL algorithm could not converge to a single policy. However, certain states in the Qtable of the AfflogRL tutor after training, display clear enough results that demonstrate what the RL agent has learned.

For example, in the zero state, where no tutorial is yet taught, the system has two choices: Either show tutorial tut21 or tutorial tut30 to the user. tut21 is a 10 minute video tutorial that gives instructions on how to play the game. This tutorial covers the first 3 chapters of the course, and after it, follows 3 tests (1 for each chapter to verify whether the user has understood that chapter). tut30 is a simple tutorial with a text and a picture that only covers the first section of the first chapters. The Q Value for tut21, is 0.19259, while the Q Value for tut30 is 24.91952. These values show that the RL agent will always start the tutoring session with tutorial 30 while choosing to follow a max Q value policy. This means that the users who watched the video, either were really bored or angry when the video ended (which is not the case) or they failed some of the tests presented after it, which is what actually happened. The reason behind this could be that by watching the ten minute video, many of the users were overwhelmed by the volume of information and forgot the answers to some of the early tests (that could be found in the second or third minute). In contrast, users that started with tut30, when reaching the first test, have only learned how to set up the game which is quite a straightforward task, and are more likely to answer that test correctly.

In another example in the state 101, where only two tutorials have been taught, (tut30 and tut31), the policy values show that the system should either teach one of the next tutorials (Qvalue = 23.9314) or perform a praise action (Qvalue = 14.12597412602021). Praise actions are only allowed by the action selector if the current emotional state is positive with a strong valence. As we saw in chapter 4, a feeling can become stronger if a user clicks the same feeling in successive tutorials. So, since the first tutorials are quite easy, there is a great possibility that the users will display a positive emotion. Hence the high Qvalue for praise. However, the majority of users may display different feelings at the start of the course, and so will prefer to move on to the next tutorial.

A more advanced state such as state 111, can also show if a test of a chapter is preferred than another. In this state, the user has been presented to all of the tutorials of the second chapter, and the RL agent has to choose the next action. At that point, the system can choose either test tst03, test tst04, or offer praise. The respective Q-values for this state are: 23.3442, 8.06, and 7.2936 , so the Agent shows a clear preference for tst03 over tst04. Now let us examine these two tests. Both of them are multiple choice questions. Tst03 text reads as:

“Can a settlement be placed on an intersection adjacent to another intersection?

(A) Yes. (B) Yes, Conditionally. (C) No.”

The correct answer which is clearly stated in chapter 2 is C. The majority of users seem to have answered this question correctly, although some users got confused and went for answer B.

tst04 reads as : “What is the minimum number of resource cards that you can have in the beginning?

(A) 3. (B) 5. (C) 6.”

This is a trickier question with the correct answer being B “5”. The correct answer is not stated in any part of the second chapter, and the user has to infer it using what he learned in this chapter. Many users failed this test, leaving some of them confused or frustrated and since the RL agent is driven by the user’s progress and positive and negative affects it will favor the more straightforward test03 over tst04.

As far as learning gains are concerned, first, the high values of both systems' learning gains may be attributed to the fact that the majority of the users were oblivious to the use-case course before learning it. Thus their very low scores scored at pre-tests. A different course containing a wide variety of topics would give the opportunity for users to score more on pre-test and thus limit their learning gains. In order to verify this, one should create and evaluate different courses.

Another problem is the relative difficulty of the pre-tests and post-tests. If the pre-tests are easy, then the learning gains would be lesser as the difference between pre-tests and post-tests would be narrower. One solution to that would be to evaluate pre-tests and post-tests in order to see if they represent the course correctly.

Learning gains may also depend on the relationship between different user groups and the course subject. The users that performed this evaluation were adult men and women of ages 20 to 72. Different clusters of this user group divided by age groups, or interests may have fared better or worse in the Settlers of Catan course.

Chapter 7
Conclusion

This doctoral dissertation describes the work of combining the advantages of different educational systems in order to create an Artificial-Intelligence-driven course-teaching system suitable for MOOCs. Specifically, we combined the architecture of an Affective Tutoring system with the adaptation mechanism of an Adaptive Learning system in order to build a domain agnostic tutoring system that can adapt to the user according to the users needs. That system can also give emotional feedback to the user and also test their knowledge.

First, we designed and implemented Afflog, an Affective tutoring system that could use Answer Set Programming [2] and the Event calculus action language [3] to represent and create courses out of a learning material pool, adapted to the learning style of the user. In order to represent learning styles we used the widely used Felder - Silverman model [1].

Second, due to wide criticism of the validity of learning styles from the scientific community we designed and implemented Afflog RL, an alternative version of the first system, which instead of adapting to the learning style of the user, made use of a Reinforcement Learning approach in order to adapt to the preferences of the user. This RL task was guided by the affective cues of the user and their relative progress in the course, through the reward function. The resulting system was then trained by 101 users who participated in a use-case course on how to play the “Settlers of Catan” board game.

Finally, we evaluated both systems using the measure of learning gains [8] that can measure new knowledge gained from a user after their interaction with the system. Evaluations showed that users displayed high learning gains after interacting with either of the systems. However, the average learning gain of both systems were similar with the Afflog RL system performing marginally better.

7.1 Validation of Research Hypotheses and contributions

We have shown that KR methods such as ASP and the Event Calculus can be used in order to create a formal model and implementation for a Tutoring System. We also proved that it is feasible to model the teaching session as a Markov Decision Process (MDP) using the same ASP and Event Calculus constructs. We also learned that The Felder-Silverman learning style model, as well as the emotions and affective strategies that are proposed in this work are suitable to be used for the adaptive, and affective parts of the Afflog system.

Although we have developed a Reinforcement Learning Tutor in order to optimize the system’s decision making process and thus utilize the best teaching strategies, our evaluation was inconclusive since Afflog RL performed similarly to Afflog. Surely, more work is needed in the design of the RL agent as well as extensive evaluation.

The research contribution of this work is that to our knowledge, this is the first ATS system that utilizes Knowledge Representation (KR) methods, and also attempts to create a course teaching system that combines the advantages of ITS, ATS and Adaptive Learning Systems. Specifically, the planning, projection, and selection methods demonstrate how known KR methods can be used in a new domain such as - but not restricted to - teaching.

Also, this work suggests that given enough training samples and optimization, a RL driven system that uses a similar architecture will perform better than an adaptive Learning System. Another contribution is the utilization of the user's emotional state as a reward function in an RL-driven ATS.

7.2 Further discussion on results and future work

7.2.1 Courses

Although the use-case course produced high learning gains in the evaluations, there are many open problems regarding possible courses and their structure. First and foremost, the introduction of other courses usable by the Afflog programs will further evaluate the course structure of the course as well as the learning gains of the users, giving us a better look on the efficiency of the programs. These courses should have certain features that the “Settlers of Catan” course lacked.

First, these courses should be larger, with more chapters, sections and a greater number of tutorials and tests. However, a large course will increase the runtime of the ASP programs. The question is will that runtime be acceptable or will some ASP programs require changes in order to be more efficient? In chapter 6 all ASP programs except the planners ran in less than a second. If that runtime increases to 2 seconds, then optimizations should be made. Another problem is the number of the tutorials available. In the “Settlers of Catan” course a number of sections had only one tutorial starting from that section, giving the planner and the action limiter programs a limited choice regarding choosing tutorials for these sections. Also, there should be more tests for each chapter in order to minimize the odds of repeating the same test.

Second, besides volume, there should be a greater variety of tutorials and tests. In the use-case course, we had text tutorials, pictures, audio tutorials of a person reciting the text and a single video describing all the course. More videos should be created, as many users found the video easier to follow. Also, in the use-case all tutorials just stated facts and did not try to challenge the user by presenting them with problems or exercises for them to solve. Similarly, the tests of the use-case were composed only of a simple multiple choice question that did not require justification from the user. A STEM course for example, should contain an equal amount of exercise tutorials and fact tutorials and have a variety of tests with multiple questions including exercises and answers with some form of justification.

Third, different topics should be explored. The use-case course was an instruction manual on how to play a board game. It was primarily selected because it was a lighter subject than, for example, a math course, also it did not require any a priori knowledge from the users, and was not a subject restricted to any age or social group. However, since it did not convey much critical or useful

knowledge and skills aside from learning how to play the boardgame, it may not have been motivating enough for some users making it harder for them to achieve the "flow" state. Therefore, any future courses should convey more practical skills, and be targeted to a specific group of users, either defined by their age, their specialization, their interests, etc. This does not mean however that the concept of gamification should be abandoned. As serious games are often more effective than traditional teaching. Also, a sequence of new courses, where the second continues where the first one left off,would ensure prior knowledge in the second course's pretest and thus give a more realistic learning gain score.

7.2.2 Reinforcement Learning

The two main problems with the Reinforcement Learning approach used in chapter 6, was the lack of sufficient training data available and the lack of adaptation to a single user. Although 101 users were used to train the system, the algorithm did not manage to converge to a single policy. That is, after training, the Afflog RL system kept exploring the state-space during the evaluation stage by selecting different tutorials for users that were in the same state. That signifies that more training episodes should have been done before Afflog RL is able to teach a group of users. The lack of adaptation to a single user is an even greater problem. Even if the current system was trained by a single user for enough episodes, it would only find the optimal policy to teach the user that specific course, as this policy would be useless to other courses. In order to solve these problems we have thought about some future work:

On the surface, it would appear that more courses would help with the training of the reinforcement learning agent. A single user completing a number of courses would enable AfflogRL to adapt to that user's way of learning, rather than to adapt to the learning habits of a group of users. However, since we have modeled each tutorial and test as a different action and each state as a sum of the results of these actions, this would result in a different Q-table for each course where each Q-table would be updated only during interaction of the user with its respective course. In order for multiple courses to be able to help with reinforcement learning training, there is a need to optimize Q-value updates and also manage to transfer knowledge from one Q-table to another. Relational RL [7] techniques could be used to optimize Q-value update, and also update Q-values from all Q-tables at the same time,by measuring a similarity score between actions and between states.

Another way to train the reinforcement learning system would be simulated students [9].]By simulating the behavior and learning processes of students, it is possible to generate unlimited data for training RL tutoring systems. However, this approach presents its own set of challenges such as the validity and complexity of simulated users.

7.2.3 Learning styles and emotion detection

Even if there has not been sufficient scientific evidence to support the existence of learning styles, they are still being used extensively by the e-learning community. Different learning styles could be encoded to Afflog's student model, and through evaluations using different courses, an optimal learning style could be found.

As we described in chapter 1, we have not used any emotion detection techniques, since it is still a growing field, and a combination of different sensors are required in order to detect general feelings, sensors that can be uncomfortable and intrusive for the students. Moreover, many AI education experts doubt the ability of current emotion detection devices to recognize the user's emotional state while learning since each person expresses themselves differently during learning. As the field progresses it would be interesting to have Afflog react to the actual emotions of its users, as the current input of one's emotions can be heavily biased.

References

1. Richard M Felder, and Linda K Silverman. Learning and teaching styles in engineering education. *Engineering education*, 78(7):674-681, 1988.
2. Michael Gelfond, and Vladimir Lifschitz. The stable semantics for logic programs. In *Proceedings of the 5th International Conference on Logic Programming,* pages 1070-1080, 1998.
3. Robert Kowalski, and Marek Sergot. A logic-based calculus of events. In *Foundations of knowledge base management*, pages 23-55. Springer, 1989.
4. Harold Pashler, Mark McDaniel, Doug Rohrer, and Robert Bjork. Learning styles: Concepts and evidence. In *Psychological science in the public interest*, 9(3):105-119, 2008.
5. Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Philipp Wanko. Theory solving made easy with clingo 5. In *Technical Communications of the 32nd International Conference on Logic Programming*, ICLP 2016), 2016, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
6.
7. Tae-Won Kim, Joohyung Lee, and Ravi Palla. Circumscriptive event calculus as answer set programming. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.
8. Dvzeroski Sasvo, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. In *Machine learning*, 43(1):7-52, 2001, Springer.
9. Cecile H. McGrath, Benoit Guerin, Emma Harte, Michael Frearson, and Catriona Manville. Learning gain in higher education. *Santa Monica, CA: RAND Corporation*, 2015.
10. Jonathan Rowe, Bob Pokorny, and Benjamin Goldberg, Bradford Mott, and James Lester. Toward simulated students for reinforcement learning-driven tutorial planning in GIFT. In *Proceedings of R. Sottilare (Ed.) 5th Annual GIFT Users Symposium*. Orlando, FL, 2017.
11. Benjamin S Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, 13(6):4-16, 1984, Sage Publications Sage CA: Thousand Oaks, CA.
12. Albert Corbett. Cognitive computer tutors: Solving the two-sigma problem. In *International Conference on User Modeling*, pages 137-147, 2001,Springer.
13. Leslie J Briggs, Peggie L Campeau, Robert M Gagne, Mark A May. Instructional media: A procedure for the design of multi-media instruction. *A Critical Review of Research, and Suggestions for Future Research*, 1967.
14. Robert M Gagne. The analysis of instructional objectives for the design of instruction. *Teaching machines and programmed learning II: Data and directions*,21-65, 1965, ERIC.
15. Timothy A Carey, and Richard J Mullan. What is Socratic questioning? *Psychotherapy: theory, research, practice, training*, 4(3):217, 2004, Educational Publishing Foundation.
16. Paul T Baffes. *Automatic student modeling and bug library construction using theory refinement*, The University of Texas at Austin, 1994.
17. Ira P Goldstein. The genetic graph: a representation for the evolution of procedural knowledge. *International Journal of Man-Machine Studies*, 11(1):51-77, 1979, Elsevier.

18. Valerie J Shute, and Joseph Psotka. Intelligent Tutoring Systems: Past, Present, and Future. ARMSTRONG LAB BROOKS AFB TX HUMAN RESOURCES DIRECTORATE, 1994.
19. KR Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*:603-649, 2020, Springer.
20. Max Kuhn, Kjell Johnson. *Applied predictive modeling*, 26, Springer, 2013.
21. Neelu Jyothi Ahuja, and Roohi Sille. A critical review of development of intelligent tutoring systems: Retrospect, present and prospect. *International Journal of Computer Science Issues (IJCSI)*, 10(4): 39, 2013, Citeseer.
22. Albert Corbett. Cognitive computer tutors: Solving the two-sigma problem. *International Conference on User Modeling*, pages 137-147, 2001, Springer.
23. Peter A Cohen, James A Kulik, Chen-Lin C Kulik. Educational outcomes of tutoring: A meta-analysis of findings. *American educational research journal*,19(2):237-248, 1982, Sage Publications.
24. Benjamin S Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, 13(6):4-16, 1984, Sage Publications Sage CA: Thousand Oaks, CA.
25. Hyacinth S Nwana. Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, 4(4):251-277, 1990, Springer.
26. Shanky Sharma, Snehal Ghorpade, Anikit Sahni, and Niti Saluja. Survey of Intelligent Tutoring Systems: a review on the development of expert/intelligent tutoring systems, various teaching strategies and expert tutoring system design suggestions. *International Journal of Engineering Research \& Technology,* 3(11):37-42, 2014.
27. Neha Mendjoge, Abhijit R Joshi, and Meera Narvekar. Review of knowledge representation techniques for Intelligent Tutoring System. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2508-2512, 2016, IEEE.
28. Antonija Mitrovic, and Kenneth R Koedinger, and Brent Martin. A comparative analysis of cognitive tutoring and constraint-based modeling. *In International Conference on User Modeling*, pages 312-322, 2003 , Springer.
29. Kurt VanLehn, Collin Lynch, Kay Schulze, Joel A Shapiro, Robert Shelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill. The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education,*15(3):147-204, 2005, IOS Press.
30. Stellan Ohlsson. Learning from performance errors. *Psychological review*, 103(2):241, 1996, American Psychological Association.
31. Scotty Craig, Arthur Graesser, Jeremiah Sullins, and Barry Gholson. Affect and learning: an exploratory look into the role of affect in learning with AutoTutor. *Journal of educational media,* 29(3):241-250, 2004, Taylor & Francis.
32. Sidney D'mello, and Art Graesser. AutoTutor and affective AutoTutor: Learning by talking with cognitively and emotionally intelligent computers that talk back. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(4):1-39, 2013, ACM New York, NY, USA.
33. Diane Litman, and Scott Silliman. ITSPOKE: An intelligent tutoring spoken dialogue system. In *Demonstration papers at HLT-NAACL 2004*, pages 5-8, 2004.
34. W Lewis Johnson, and Andre Valente. Tactical language and culture training systems: Using AI to teach foreign languages and cultures. *AI magazine*, 30(3):72, 2009.
35. Benedict Du Boulay, and Rosemary Luckin. Modelling human teaching tactics and strategies for tutoring systems: 14 years on. *International Journal of Artificial Intelligence in Education*, 26(1): 393-404, 2016, Springer.
36. Benedict du Boulay, Rosemary Luckin, and Teresa del Soldato. The plausibility problem: Human teaching tactics in the ‘hands’ of a machine. In *Artificial intelligence in education, open learning environments: New computational technologies to support learning, exploration, and collaboration, proceedings of AIED-99,* pages 225-232, 1999, IOS Press Amsterdam.

37. John . R. Anderson. The adaptive character of thought. Hillsdale, NJ: Lawrence ErlbaumAssociates. *Inc. AndersonThe adaptive character of thought1990* , 1990.
38. John R Anderson, Albert T Corbett, Kenneth R Koedinger, and Ray Pelletier. Cognitive tutors: Lessons learned. *The journal of the learning sciences*, 4(2):167-207, 1995, Taylor & Francis.
39. Kurt VanLehn, Arthur C Graesser, G Tanner Jackson, Pamela Jordan, Andrew Olney, and Carolyn P Rose. When are tutorial dialogues more effective than reading? *Cognitive science*, 31(1):3-62 , 2007, Wiley Online Library.
40. David Wood, Heather Wood, and David Middleton. An experimental evaluation of four face-to-face teaching strategies. *International journal of behavioral development*, 1(2):131-147, 1978, Sage Publications Sage CA: Thousand Oaks, CA.
41. John R Anderson. ACT: A simple theory of complex cognition. *American psychologist*, 51(4):355, 1996, American Psychological Association.
42. Gordon Pask, BCE Scott, and D Kallikourdis. A theory of conversations and individuals (exemplified by the learning process on CASTE. *International Journal of Man-Machine Studies,* 5(4):443-566, 1973, Elsevier.
43. Tak-Wai Chan, and Chih-Yueh Chou. Exploring the design of computer supports for reciprocal tutoring. *International Journal of Artificial Intelligence in Education (IJAIED)*, 8:1-29, 1997.
44. Michelene TH Chi, Miriam Bassok, Matthew W Lewis, Peter Reimannand, Robert Glaser. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive science*, 13(2):145-182, 1989, Elsevier.
45. Allyson Fiona Hadwin, and Philip H Winne. CoNoteS2: A software tool for promoting self-regulation. *Educational Research and Evaluation*, 7(2-3):313-334, 2001, Taylor & Francis.
46. Ivon Arroyo, Joseph E Beck, Beverly Park Woolf, Carole R Beal, and Klaus Schultz. Macroadapting Animalwatch to gender and cognitive differences with respect to hint interactivity and symbolism. In *International conference on intelligent tutoring systems*, page 574-583, 2000, Springer.
47. Syeda Erfana Zohora, AM Khan, Arvind K Srivastava, Nhu Gia Nguyen, and Nilanjan Dey. A study of the state of the art in synthetic emotional intelligence in affective computing. *International Journal of Synthetic Emotions (IJSE),* 7(1):1-12, 2016, IGI Global.
48. Barbara Mandell, and Shilpa Pherwani. Relationship between emotional intelligence and transformational leadership style: A gender comparison. *Journal of business and psychology*, 17(3):387-404, 2003 , Springer.,
49. Peter Salovey, and John D Mayer. Emotional intelligence. *Imagination, cognition and personality*, 9(3):185-211, 1990, Publications Sage CA: Los Angeles, CA.
50. Rosalind W Picard. *Affective computing,* MIT press, 2000.
51. Jeanne Nakamura, and Mihaly Csikszentmihalyi. Flow theory and research, *Handbook of positive psychology*, 195:206, 2009.
52. Derek McColl, Alexander Hong, Naoaki Hatakeyama, Goldie Nejat, and Beno Benhabib. A survey of autonomous human affect detection methods for social robots engaged in natural HRI. *Journal of Intelligent & Robotic Systems*, 82(1):101-133, 2016, Springer.
53. Aleix Martinez, and Shichuan Du. A model of the perception of facial expressions of emotion by humans: research overview and perspectives. *Journal of Machine Learning Research*, 13(5), 2012.
54. Didier Grandjean, David Sander, and Klaus R Scherer. Conscious emotional experience emerges as a function of multilevel, appraisal-driven response synchronization. *Consciousness and cognition*, 17(2):484-495, 2008, Elsevier.
55. James A Russell. Evidence of convergent validity on the dimensions of affect. *Journal of personality and social psychology*, 36(10):1152, 1978 , American Psychological Association.
56. Albert Mehrabian. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, 14(4):261-292, 1996, Springer.
57. Johnny RJ Fontaine, Klaus R Scherer, Etienne B Roesch, and Phoebe C Ellsworth. The world of emotions is not two-dimensional. *Psychological science*, 18(12):1050-1057, 2007, Sage Publications Sage CA: Los Angeles, CA.

58. Gary McKeown, Michel F Valstar, Roderick Cowie, and Maja Pantic. The SEMAINE corpus of emotionally coloured character interactions. In *2010 IEEE International Conference on Multimedia and Expo*, pages 1079-1084, 2010, IEEE.
59. Paul Ekman, Wallace V Friesen, and Phoebe Ellsworth. *Emotion in the human face: Guidelines for research and an integration of findings*, 11. Elsevier, 2013.
60. Emiliano Lorini, and Francois Schwarzentruber. A logic for reasoning about counterfactual emotions, *Artificial Intelligence,* 175(3-4):814-847, 2011, Elsevier.
61. Bas R Steunebrink, Mehdi Dastani and John-Jules Ch Meyer. A formal model of emotion triggers: an approach for BDI agents. *Synthese*, 185(1):83-129, 2012, Springer.
62. Felix D Schonbrodt, and Jens B Asendorpf. The challenge of constructing psychologically believable agents. *Journal of Media Psychology: Theories, Methods, and Applications*, 23(2):100, 2011, Hogrefe Publishing.
63. Frank Van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of knowledge representation*, Elsevier, 2008.
64. Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331-434, 1990, Elsevier.
65. Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237-285, 1996.
66. Huong May Truong. Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities. Computers in human behavior, 55:1185-1193, 2016, Elsevier.
67. Klaus Teuber. Settlers of Catan, 2007, Mayfair Games.
68. Christopher JCH Watkins, and Peter Dayan. Q-learning. *Machine learning*, 8(3):279-292, 1992, Springer.
69. Martin Schrepp. User experience questionnaire handbook, *All you need to know to apply the UEQ successfully in your project*, 2015.
70. Edeh Michael Onyema, and Eucheria, Nwafor Chika. Impact of Coronavirus pandemic on education. *Journal of education and practice*, 11(13):108-121, 2020, USA.
71. microsoft teams website. https://www.microsoft.com/el-gr/microsoft-teams/group-chat-software, 9 - 5 - 2022, Microsoft.
72. zoom. https://zoom.us, 9-5-2022.
73. Derek Sleeman, and John Seely Brown. *Intelligent tutoring systems*, London: Academic Press, 1982.
74. Eleanor O'Rourke, Eric Butler, Tolentino Armando Diaz, and, Zoran Popovic. Automatic generation of problems and explanations for an intelligent algebra tutor. In *International Conference on Artificial Intelligence in Education*, pages 383-295, 2019, Springer.
75. Nik Thompson, and Tanya Jane McGill. Genetics with Jean: the design, development and evaluation of an affective tutoring system. *Educational Technology Research and Development*, 65(2):279-299, 2017, Springer.
76. Sam Alexander, Abdolhossein Sarrafzadeh, and Chao Fan. Pay attention! The computer is watching: Affective tutoring systems. In *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, pages 1463-1466, 2003, Association for the Advancement of Computing in Education (AACE).
77. Alexandros Paramythis, and Susanne Loidl-Reisinger. Adaptive learning environments and e-learning standards. In *Second european conference on e-learning*, 1, pages 369-379, 2003.
78. Abram Anders. Theories and applications of massive online open courses (MOOCs): The case for hybrid design. *International Review of Research in Open and Distributed Learning*, 16(6):39-61, 2015, Athabasca University Press (AU Press).
79. Anthony Stentz. focussed d^* algorithm for real-time replanning. In IJCAI, 95, 1652-1659, 1995.
80. Robert A Sottilare, Keith W Brawner, Benjamin S Goldberg, and Heather K Holden.The generalized intelligent framework for tutoring (GIFT). *Orlando, FL: US Army Research Laboratory-Human Research & Engineering Directorate (ARL-HRED)*, 2012.

81. Jakob Nielsen, and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249-256, 1990.
82. Eleni Papadaki, Xenophon Zabulis, Stavroula Ntoa, George Margetis, Panagiotis Koutlemanis, Polykarpos Karamaounas, and Constantine Stephanidis. The book of Ellie: An interactive book for teaching the alphabet to children. In *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1-6, 2013, IEEE.
83. Research ethics committee of the University of Crete, https://en.uoc.gr/research-at-uni/ethics, 9-5-2022.
84. Richard S Sutton, and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
85. Xian Shun Chen. chen0040/java-reinforcement-learning. https://github.com/chen0040/java-reinforcement-learning, 20-1-2023.
86. Lei Bao.Theoretical comparisons of average normalized gain calculations. *American journal of physics,* 74(10):917-922, 2006, American Association of Physics Teachers.
87. Richard R Hake. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American journal of Physics,* 66(1):64-74, 1998, American Association of Physics Teachers.
88. discord, Your place to talk and hang out, https://discord.com/, 20 - 1 - 2023.
89. Anydesk, the fast remote desktop application. https://anydesk.com, 20-1 -2022.
90. Messenger. https://messenger.com, 20 -1 -2022.
91. Skype. https://skype.com, 20 -1 -2022.
92. Michael W Eysenck, Nazanin Derakshan, Rita, Santos, and Manuel Calvo. Anxiety and cognitive performance: attentional control theory. *Emotion*, 7(2):336, 2007, American Psychological Association.
93. Norbert Schwarz, and Gerald L Clore. Mood, misattribution, and judgments of well-being: informative and directive functions of affective states. Jou*rnal of personality and social psychology*, 45(3):513, 1983, American Psychological Association.
94. Henry C Ellis. Resource allocation model of the effect of depressed mood states on memory. *Affect, cognition, and social behavior* , 1988, Hogrefe.
95. Norbert Schwarz. *Feelings as information: Informational and motivational functions of affective states*. The Guilford Press, 1990.
96. Daniel Goleman. *Emotional intelligence: Why it can matter more than IQ*. Bloomsbury Publishing, 1996.
97. Affectiva Media Analytics. https://go.affectiva.com/affdex-for-market-research, 25 - 1 -2022.
98. Harald G Wallbott. Bodily expression of emotion. *European journal of social psychology*, 28(6), 1998, Wiley Online Library.
99. Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia* 19(2):4-10, 2012, IEEE.
100. Klaus R Scherer. Vocal communication of emotion: A review of research paradigms. *Speech communication*, 40(1-2):227-256, 2003, Elsevier.
101. Martijn Goudbeek, and Klaus Scherer. Beyond arousal: Valence and potency/control cues in the vocal expression of emotion. *The Journal of the Acoustical Society of America*, 128(3):1322-1226, 2010, Acoustical Society of America.
102. Hatice Gunes, Bjorn Schuller, Maja Pantic, and Roddy Cowie. Emotion representation, analysis and synthesis in continuous space: A survey. In *2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG)*, pages 827-834, 2011, IEEE.
103. Florian Eyben, Martin Wollmer, and Bjorn Schuller. OpenEAR—introducing the Munich open-source emotion and affect recognition toolkit. In *2009 3rd international conference on affective computing and intelligent interaction and workshops*, pages 1-6, 2009, IEEE.
104. Klaus R Scherer, and Tanja Banziger.Emotional expression in prosody: a review and an agenda for future research, In *Speech prosody 2004, international conference*, 2004.

105. Osman Hussein Al, and Tiago H Falk. Multimodal affect recognition: Current approaches and challenges. *Emotion and attention recognition based on biological signals and images*, 59-86, 2017, IntechOpen London, UK.
106. Roddy Cowie, Ellen Douglas-Cowie, Margaret McRorie, Ian Sneddon, Laurence Devillers, and Noam Amir. Issues in data collection. *Emotion-Oriented Systems: The HUMAINE Handbook*, 197-212, 2011, Springer.
107. Michael Grimm, Kristian Kroschel, and Shrikanth Narayanan. The Vera am Mittag German audio-visual emotional speech database. In 2*008 IEEE international conference on multimedia and expo*, pages 865- 868, 2008, IEEE.
108. Jing Chen, Chenhui Wang, and Kejun Wang. HEU Emotion: a large-scale database for multimodal emotion recognition in the wild. *Neural Computing and Applications*, 33:8669-8685, 2021, Springer.
109. Barry Kort, Rob Reilly, and Rosalind W Picard. An affective model of interplay between emotions and learning: Reengineering educational pedagogy-building a learning companion. In *Proceedings IEEE international conference on advanced learning technologies*, pages 43-46, 2001, IEEE.
110. Jan L Plass, Steffi Heidig, Elizabeth O Hayward, Bruce D Homer, and Enjoon Um. Emotional design in multimedia learning: Effects of shape and color on affect and learning. *Learning and Instruction*, 29:128-140, 2014, Elsevier.
111. Eunjoon Um, Jan L Plass, Elizabeth O Hayward, Bruce D Homer and others. Emotional design in multimedia learning. *Journal of educational psychology*, 104(2):485, 2012, American Psychological Association.
112. Beverly Woolf, Winslow Burleson, Ivon Arroyo, Toby Dragon, David Cooper, and Rosalind Picard. Affect-aware tutors: recognising and responding to student affect. *International Journal of Learning Technology*, 4(3-4):129-164, 2009, Inderscience Publishers.
113. Hector Levesque, Fiora Pirri, and Ray Reiter. *Foundations for the situation calculus*, oping University Electronic Press, 1998.
114. Jinying Lin, Zhen Ma, Randy Gomez, Keisuke Nakamura, Bo He, and Guangliang Li. A review on interactive reinforcement learning from human social feedback. *IEEE Access*, 8:120757 - 120765, 2020, IEEE.
115. Peter Honey, and Alan Mumford. *The manual of learning styles*, Peter Honey, 1986.
116. Tzu-Chi Yang, Gwo-Jen Hwang and Stephen Jen-Hwa Yang. Development of an adaptive learning system with multiple perspectives based on students' learning styles and cognitive styles. *Journal of Educational Technology & Society*, 16(4):185-200, 2013, JTOR.
117. Shaimaa M Nafea, and Franccois Siewe. On recommendation of learning objects using felder-silverman learning style model. *IEEE Access*, 7:163034-163048, 2019, IEEE.
118. Diana Zagulova, Viktorija Boltunova, Sabina Katalnikova, Natalya Prokofyeva, and Kateryna Synytsya. Personalized E-Learning: Relation Between Felder-Silverman Model and Academic Performance. *Appl. Comput. Syst*.,24(1):25-31, 2019.
119. Dragan Zlatkovic, Nebojsa Denic, and Milena Petrovic. Analysis of adaptive e-learning systems with adjustment of Felder-Silverman model in a Moodle DLS. *Computer Applications in Engineering Education*, 28(4):803-813, 2020,Wiley Online Library.
120. Paul A Kirschner. Stop propagating the learning styles myth. *Computers & Education*, 106: 166-171, 2017, Elsevier.
121. Cedar Riener, and Daniel Willingham. The myth of learning styles. *Change: The magazine of higher learning*, 42(5):32-35, 2010, Taylor & Francis.
122. Ivan J Prithishkumar, Stelin Agnes Michael, and others. Understanding your student: Using the VARK model. *Journal of postgraduate medicine*, 60(2):183, 2014, Medknow Publications.
123. Eugene Sadler-Smith. ‘Learning style’: frameworks and instruments. *Educational psychology*, 17(1-2):51-63, 1997, Taylor & Francis.

124. Dade Nurjanah. Good and Similar Learners' Recommendation in Adaptive Learning Systems, page 434-440, 2016.
125. Kallirroi Georgila, Mark G Core, Benjamin D, Nye, Shamya Karumbaiah, Daniel Auerbach, and Maya Ram. Using reinforcement learning to optimize the policies of an intelligent tutoring system for interpersonal skills training. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, page 737-745, 2019.
126. Eleni Fotopoulou, Anastasios Zafeiropoulos, Michalis Feidakis, Dimitrios Metafas, and, Symeon Papavassiliou. An interactive recommender system based on reinforcement learning for improving emotional competences in educational groups. In *Intelligent Tutoring Systems: 16th International Conference, ITS 2020, Athens, Greece, June 8--12, 2020, Proceedings 16*, pages 248-258, 2020, Springer.
127. Min Chi, Kurt VanLehn, Diane Litman, and Pamela Jordan. An evaluation of pedagogical tutorial tactics for a natural language tutoring system: A reinforcement learning approach. *International Journal of Artificial Intelligence in Education*, 21(1-2_:83:113, 2011, IOS Press.
128. Hae Won Park, Ishaan Grover, Samuel Spaulding, Louis Gomez, and Cynthia Breazeal. A model-free affective reinforcement learning approach to personalization of an autonomous social robot companion for early literacy education. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):687-694, 2019.
129. Matthias Nickles, and Achim Rettinger. Interactive relational reinforcement learning of concept semantics. *Machine learning*, 94:169-204, 2014, Springer.
130. Djananjay Pandit, and Ajay Bansal. A declarative approach for an adaptive framework for learning in online courses. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, 1, pages 212-215, 2019 , IEEE.
131. Cheng-Hung Wang, and Hao-Chiang Koong Lin. Constructing an affective tutoring system for designing course learning and evaluation. *Journal of Educational Computing Research*, 55(8):1111-1128, 2018 , SAGE Publications Sage CA: Los Angeles, CA.
132. Samy S Abu Naser, ITSB: an intelligent tutoring system authoring tool, 2016.
133. Kenneth R Koedinger, Vawmm Aleven and Neil Heffernan. Toward a rapid development environment for cognitive tutors. In *12th Annual Conference on Behavior Representation in Modeling and Simulation*, 2003.
134. Mohanad iles, and Massoud Agha. Knowledge-based ITS for Teaching Mongo Database, 2017.
135. Hani M Sh Bakeer, and A Samy S bu-Naser. An intelligent tutoring system for learning TOEFL, 2019.
136. Abed Elhaleem A Elnajjar, and Samy S Abu Naser. DES-Tutor: An Intelligent Tutoring System for Teaching DES Information Security Algorithm, 2017.
137. Hasan A Abu Hasanein, and Samy S Abu-Naser. An intelligent tutoring system for cloud computing, 2017.
138. Shu-Chuan, Shih, Chih-Chia Chang, Bor-Chen, Kuo, and Yu-Han Huang. Mathematics intelligent tutoring system for learning multiplication and division of fractions based on diagnostic teaching.*Education and Information Technologies*, 1-22, 2023, Springer.
139. Roger Nkambou, Janie Brisson, Clauvice Kenfack, Serge Robert, Pamela Kissok, and Ange Tato. Towards an intelligent tutoring system for logical reasoning in multiple contexts. In *Design for Teaching and Learning in a Networked World: 10th European Conference on Technology Enhanced Learning, EC-TEL 2015, Toledo, Spain, September 15-18, 2015, Proceedings 10* , 460-466, 2015, Springer.
140. Nik Thompson, and Tanya Jane McGill. Affective stack—a model for affective computing application development. *Journal of Software*, 10(8):919-930, 2015, Academy Publisher.
141. Arturas Kaklauskas, Agne Kuzminske, and others. Affective tutoring system for built environment management. *Computers & Education*, 82:202-216, 2015, Elsevier.
142. Vincent Aleven, Bruce M McLaren, Jonathan Sewall, and Kenneth R Koedinger. The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In *Intelligent Tutoring*

Systems: 8th International Conference, ITS 2006, Jhongli, Taiwan, June 26-30, 2006. Proceedings 8, pages 61-70, 2006, Springer.

Appendix A
Publications and Systems

Work from this dissertation was published in two conferences:

1 Achilles Dougalis, and Dimitris Plexousakis. AFFLOG: A Logic Based Affective Tutoring System. In *Intelligent Tutoring Systems: 16th International Conference.* ITS 2020*,* pages 270-274, Athens, Greece, 2020. Springer.

2 Achilles Dougalis, and Dimitris Plexousakis, A Logic Based Affective Tutoring System that uses Reinforcement Learning for discovering Teaching Strategies. In *EDULEARN22 Proceedings*. EDULEARN22, pages 4535-4543, Palma, Spain, 2022 IATED.

Two systems were created: Afflog, and Afflor RL.

117

Appendix B
Acronyms

ASP : Answer Set Programming
DEC: Discrete Event Calculus
RL : Reinforcement Learning
TD : Temporal Difference

119

Appendix C
Evaluation Forms

Settlers of Catan PostTest and PreTest

For the Pretest and post test of the settlers of Catan, we used Google forms with the following multiple choice questions:

1. How many Victory Points do you need in order to win the game?

    A. 10
    B. 20
    C. 15 Correct answer.

2. What is the name of the token that is placed on a hex to prevent resources from that hex being distributed?

    A. The Leader
    B. The Knight
    C. The Robber Correct answer.

3. What is the correct way of placing the Hexes?

    A. Place all the hexes face down. Correct answer.
    B. The desert hex in the middle and there are not two hexes adjacent to one another.
    C. Both of the above are correct.

4. What is the minimum number of resource cards that you can have in the beginning?

    A. Three
    B. Five. Correct answer.
    C. Six

5. How much does it cost to build a road ?

A. One lumber, one brick. Correct answer.
B. One lumber, one brick, one grain, one wool.
C. Two grain, three ore.

6. What roll of the dice does the Robber need in order to spread mayhem in the game?

A. Snake eyes.
B. Seven. Correct answer.
C. Thirteen.

7. If a player rolls the above special number:

A. Each player with 7 or more cards must discard half of them.
B. The person who rolled gets to put the robber on whatever number token he/she desires, and then gets to take one card from any player that has a settlement or city touching the terrain hex with the robber on it.
C. Both are correct. Correct answer.

8. Can a settlement be placed on an intersection adjacent to another intersection with a settlement?
A. Yes. Correct answer.
B. Yes, conditionally.
C. No.

9. What does the Knight Card do?

A. Move the thief to a tile of your choice.
B. Lets you get a card from a player.
C. Both of the above. Correct answer.

10. Looking to get some more points? The largest army card enters the mix after the first player uses this number of Knight cards:

A. Three.
B. Five. Correct answer.
C. Seven