This paper discusses a general framework for knowledge acquisition and management in an intelligent tutoring system. The system is based on constrained-based tutor paradigm in which the idea of “learning by performance error” is exploited. Such a theory states that in a given domain of knowledge there is a set of constraints that must be satisfied in order to provide the correct solution to the problem. In our particular case constraints are represented in the form of conditional statements (and in general as first order formula) and we aim at an easy mechanism for constraint acquisition. We propose a framework for acquiring and managing knowledge in an intelligent tutoring system and discuss the part related to the specific task of knowledge acquisition.

1. Introduction

ITSs are tools for assisted learning process. They leverage the knowledge of the domain and student models to implement adaptive tutoring that can approach the benefits of teacher to student interaction. The deployment of ITSs such as these ones [2,11,14] have been demonstrating the efficacy of this technology. The scenario is composed by two actors, the teacher and the student, each of which interacts with an ITS module. The teacher provides the necessary knowledge to the ITS by means of an authoring tool and the student gets some questions from the ITS responding with an answer. ITS gives to a student the feedback on her solution showing the errors she made. This process is iterated until the student provides the right solution.

While ITSs have been proven useful in the learning process, they suffer from the disadvantage of being difficult to setup and tune for the following reasons:
- They require an expert in knowledge engineering in order to transfer the specific domain knowledge from teacher to computer;
- Currently ITSs are still complex to setup and manage because usually they require to implement ad hoc mechanisms depending on the domain they are made for.

The proposed architecture addresses the complexity of customizing the ITSs for specific knowledge domains by partitioning the knowledge codification and manipulation tasks in workable subtask some of which can be aided by more advanced modules as the related research goes on. In the following section basic concepts relate to the framework are briefly reviewed, in section 3 the general architecture is described and in section 4 we discuss about the knowledge acquisition task.
2. Basic Concepts

In this paragraph we briefly introduce the main concepts involved in the knowledge acquisition framework that we combine together in order to obtain an architecture for accomplishing such task in various domains.

2.1. Constrained-based tutor

This kind of ITS is based on Ohlsson’s theory of learning from performance errors. This theory focuses the attention on the state of the solution provided by a student while resolving a problem instead of the sequence of actions needed to arrive at the correct solution. In this way the diagnostic information is supposed to be extracted from the solution state instead of student actions.

The constraint-based model proposed by Ohlsson represents both domain and student knowledge in the form of constraints, where the constraints represent the basic principles underlying the domain. A constraint is characterised by a relevance clause, and a satisfaction clause. The relevance clause is a condition that must be true before the constraint is relevant to the current solution. Once the relevance clause has been met, the satisfaction clause must be true for the solution to be correct.

For example we can consider a student learning to code with a programming language. When she has to manipulate array inside an iterative statement one of such rules can be that the index used in the array must not exceed the array dimension. A constraint for this situation might be:

\[
\text{IF array dimension is N} \\
\text{AND array index is I} \\
\text{THEN the guard is I<N}
\]

2.2. Ontology

Ontologies are used to capture knowledge about some domain of interest. Ontology describes the concepts in the domain and also the relationships that hold between those concepts. Ontologies have been proven very useful in many field of artificial intelligence and recent works in knowledge acquisition demonstrated that the use of ontology can help human expert to acquire some kind of knowledge in an intelligent tutoring system [11].

Ontologies are based on a logical model and allow some kind of integration with logical settings of logic programming [10] that can be useful in an induction process for acquiring useful information from the evidence of a given domain. In particular, with some restriction, ontology can be expressed in a form of clausal logic permitting a more efficient use of information contained in such ontology.

2.3. Inductive Logic Programming

Inductive learning is the process of doing inference of a generalized conclusion from particular instances. A specific task of inductive learning is the so called Inductive Logic Programming (ILP) in which the aim is to infer a theory in a form of logic programs starting from positive and/or negative examples of some given observations of evidence.

The general problem of ILP is formulated in the following way

- Given:
- a background knowledge $B$, as a set of Horn clauses
- a set of positive example $P$, as a set of Horn clauses
- a set of negative example $N$, as a set of Horn clauses

find an hypothesis $H$ as a set of Horn clauses such that:

- $\forall p \in P: H \cup B \models p$
- $\forall n \in N: H \cup B \not\models n$

In the following the concept of ILP will be used as a key mechanism for knowledge acquisition in our framework.

2.4. **Information Extraction from text**

The general concept of information extraction is defined as the action of finding relevant information from a great amount of text. As an example one can be interested in finding company information from an e-shopping site analyzing the article description. In our specific case we are interested in the task of identifying instances of a particular class of entity and relationship among these classes and the extraction of relevant arguments of the entity or relationship.

Information extraction is applied to natural language text but in our case we can think to apply the concept also to a piece of source code written in some programming language without loss of generality. The aim is to extract information from a text (written in some form) according to the “world description” given by an ontology.

3. **Architecture description**

The proposed architectures of tutoring and authoring module are illustrated in figures 1 and 2. We suppose a scenario where the student is asked to answer a question requiring a solution in a form of text. The solution can be expressed in natural language, for domain requiring that form, or in a programming language in case we are concerning with teaching to program a computer. The system supports the student problem solving by providing feedback that relates to student errors. Student errors are recognized using the system knowledge base which is made up of three components:

- a **domain ontology** describing the concepts and relations surrounding the domain concepts;
- a **set of parser rules** in order to find the entities corresponding to classes in the domain ontology and relationship between them;
- a **constraints base**, expressed as a first order formula, that a solution must satisfy in order to be correct.

The ontology and the parser rules can be used in two ways depending on the solution representation:

- in the case of natural language text we use an Information Extraction System (IES) to tag the various parts of solution in order to recognize the entities corresponding to the concepts and relationships in the domain ontology.
- in the case of a programming language source code we use a parser to recognize the concepts implicitly contained in a grammar and the ontology to filter the more interesting ones.
Once the text has been processed a component called “Tagged text to Prolog transducer” is used to obtain a version of the tagged solution in the form of Prolog ground facts. This set of facts must represent the information corresponding to the solution text and its structure must agree with the domain ontology.

In this form the solution can be checked against the constraint base. Following the work of [18,19] each rule in the constraint base has attached a feedback that explains the error concerning the constraint violation. This task is accomplished by Solution Checker component.

The phase of knowledge acquisition can be divided in three major tasks: ontology building, creation of the set of parser rules and construction of constraint base. We will focus on automating the construction of constraint base even if there are some methods that can be used to automate the other two tasks. The overall scheme of knowledge acquisition system is illustrated in figure 2.

The problem of finding a general rule that must be satisfied in a domain can be formulated as an Inductive Logic Programming (ILP) problem. The general setting of an ILP problem has been described in the section 2.3.

In our setting the positive, and optionally the negative, examples are given as a set of solutions to the same problem. Such solutions are then processed by the Prolog representation producer module in order to obtain a suitable representation to be processed in an ILP setting. The background knowledge is represented by the domain ontology that, with some restriction, can be transformed in a clausal logic form and by the constraint set that initially is empty. This information is processed by another component called “ILP problem generator” that produces a set of data that can be directly processed by an ILP engine.
The ILP engine will produce a new set of rules representing the constraints related to the provided examples. Such set must be validated before it can be added to the constraint database and associated to the specified problem. The whole process can be repeated until all necessary constraints are found.

4. Knowledge acquisition in constraint based tutor

In our framework we follow a slightly different setting from that of [19,20] in which a constraint deals with “ideal solution” and “student solution”. Instead we focus at finding interesting information that is hidden in the description of some examples to a problem. In particular we want to find interesting rules that are valid in the provided examples specifying the problem in various forms of ILP settings.

In this situation the various components of the problem have to be identified. Furthermore the knowledge representation has to be decided for each setting we decide to use in the knowledge discovery task. Recall the general definition of ILP problem:

- Given:
  - a background knowledge \(B\), as a set of Horn clauses
  - a set of positive example \(P\), as a set of Horn clauses
  - a set of negative example \(N\), as a set of Horn clauses
- find an hypothesis \(H\) as a set of Horn clauses such that:
  - \(\forall p \in P: H \cup B \models p\)
  - \(\forall n \in N: H \cup B \not\models n\)

The background knowledge can be surely composed by the information provided in the ontology. With some restrictions an ontology can be coded in a clausal logic form [10]. Substantially an ontology describes a graph of classes where each class can be related to another by means of relationships. We can also specify some other information but without entering in details the most
important transformations from ontology representation language to clausal logic are:

- if a class B is a subclass of class A (i.e. each object in the domain that belongs to class B belongs to class A) is expressed as: $A(X) :- B(X)$
- the same for relationships and sub-relationship: if two objects $a,b$ in the domain are related by relation $Q$ and $Q$ is sub-relation of relation $P$ then $a,b$ are in relation $Q$. This is expressed as $P(X,Y) :- Q(X,Y)$
- each relation has a range and a domain (in our case range and domain are both classes), that is a relation links objects in the range to objects in the domain

  If range of property $P$ is class $C$: $C(Y) :- P(X,Y)$;
  If domain of property $P$ is class $D$: $D(X) :- P(X,Y)$;

The positive and negative examples change depending on the setting we choose to extract interesting rules. We have considered three settings which correspond to the state of the art ILP systems:

- The system ALEPH [1] can be used to learn rules of this type: “the solution is correct is the condition $C$ is true” where $C$ is a conjunction of atoms i.e. the rules “Correct($S$) :- $A(S,Obj1), B(S,Obj2), P(S,Obj1,Obj2)$” states that solution $S$ is correct if has an object $Obj1$ of class $A$ and it has an object $Obj2$ of class $B$ and $Obj1, Obj2$ are linked by relation $P$.
  In this case the positive and negative examples are expressed as a single predicate “Correct(…)” and the description of examples is added to knowledge base

- The systems ClassicCL [17] and system Tertius [9] are able to find rules that can relate each predicate in the example with each other. In this setting, called “learning from interpretation”, the provided examples are considered as Herbrand interpretations that are models for the searched rules. In this way the found rules are true in the positive examples.
  ClassicCL is able to find rules from both only positive and positive and negative examples while Tertius accepts only positive examples for learning process. The produced rules are in this form: “if $B$ then $H$” where $B$ is a conjunction of atoms and $H$ can be a disjunction of atoms.

- WARMR [6] is a system that finds frequent queries in a deductive database. WARMR can also find the equivalent of association rules in the framework of first order logic called “query extension”. The rules are of this type: $A_1,...,A_k \Rightarrow A_{k+1},...,A_n$ where $A_i$ are atoms.
  Such rules must be interpreted in this way: “if the query $?\ A_1,...,A_k$ has a solution then the query extension $?\ A_1,...,A_n$ must have a solution”. In this case the examples are the description in the form of DATALOG clauses.

All the rules founded in the learning process must be validated by means of another module that helps the author to rank the rules. The rules that are judged general enough w.r.t. domain are added in a constraint base and used as background knowledge for successive phase of learning. All the other rules are associated with the specific problem used to generate them.
5. Conclusion

We have presented a framework for an Intelligent Tutoring System based on constraint paradigm and logic programming. This idea is inspired by the work of Mitrovic et al. used in various constraint-based ITSs [11]. The main advantage respect to other framework is that each module can be developed as a separated field of research. Once we have a representation of solution for an ILP engine we do not need to develop an ad hoc mechanism in order to find the constraints rule. The same reasoning can be applied for Information Extraction System. Furthermore the process of ontology building and parser rules creation should be automated as the research in this field go on. Some work for automatic acquisition of ontology has been done [3,8,12,16] even if they need to be improved. Another advantage is the possibility of use an open answer approach for testing instead of a strongly structured answer as in [18,19,20]. The clear separation of the stages involved in the various processes give us the strong theoretical basis of the research field involved like ILP and Information Extraction. We hope that this framework is general enough to address many type of domain knowledge even if initially we are implementing the framework for language programming that results in less difficulties respect to domains that involve natural language.

6. References


