KNOWLEDGE REPRESENTATION FOR RESTRICTION DIGESTION AND RECONCSTRUCTING DNA IN A GENETIC LAB

par

Xin Zhao

Mémoire présenté au département de mathématiques et d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, April 2004

Your file Votre référence
ISBN: 0-612-94919-2
Our file Notre référence
ISBN: 0-612-94919-2

Le 23 avril 04,
Date

le jury a accepté le mémoire de M. Xin Zhao dans sa version finale.

Membres du jury

M. André Mayers
Directeur
Département de mathématiques et d'informatique

M. Roger N'Kambou
Codirecteur
Département de mathématiques et d'informatique

M. Gabriel Girard
Membre
Département de mathématiques et d'informatique

Mme Hélène Pigot
Président-rapporteur
Département de mathématiques et d'informatique

ABSTRACT

The project issues are about knowledge representation for restriction digestion and reconstructing DNA with enzymes in the genetic lab. The research subject is to create conceptual objects and tools in a graphic interface so that the diagnosis and teaching of this knowledge are carried out easily by the tutor.

Knowledge representation is described in a document named *l'étudiant virtuel*. The document is named *l'étudiant virtuel* because it is centered on the structures of representation of knowledge used by the student and the expert. However, the purpose of these structures of representation is to describe real objects and to handle them conceptually.

The results of the thesis are establishing semantic and procedure knowledge relative to a real genetic lab. Episodic knowledge is recorded for what the student does in the genetic lab. The corresponding objects and tools have been built for real genetic lab simulation.

ACKNOWLEDGEMENTS

First and foremost, I wish to extend special thanks to my research director André Mayers. He introduced me to the fascinating world of my research work. He gave me a lot of help in my courses, my research work and the writing of this thesis. I would also like to appreciate my research co-director, Roger Nkambou, who gave a lot of help in writing the thesis, my research work and some financial support. I will always bear both of them in mind.

My acknowledgements also go to the professor, Claude Déry, who gave me a lot of information and suggestion about the biology domain. I thank my cooperator Philippe Fournier-Viger, who gave me many ideas about knowledge concepts and we have worked together to establish a knowledge base for the genetic lab. I also appreciate my friend David Gordon for checking and correcting my thesis.

I would also like to acknowledge all the professors and students who gave me some help.

CONTENTS

LIST OF ABBREVIATIONS

ASTUS	Apprentissage par Système Tutorial à l'Université de Sherbrooke
DNA	Deoxyribonucleic Acid
EJB	Enterprise JavaBeans
ER	Enzyme Restriction
ID	Identity
ITS	Intelligent Tutoring System
KB	Knowledge Base
KR	Knowledge Representation
XML	eXtensible Markup Language
XSLT	XML Stylesheet Language for Transformations

LIST OF TABLES

LIST OF FIGURES

Introduction

This project concerns designing and establishing a laboratory of conceptual experimentation for simulating restriction digestion and reconstructing DNA in the genetic lab. This lab must be integrated into the current ASTUS system. The emphasis of this project is to identify semantic and procedural knowledge, which can be used for this lab, and to reify them as objects and tools corresponding to this knowledge.

Restriction digestion and reconstructing DNA means to cut DNA into fragments with restriction enzymes and then reconstruct the DNA with the fragments. These fragments will be used to construct a restriction map, thereby confirming the identity and sequence of DNA. When a tutor teaches students how to do restriction digestion and DNA reconstruction, the students may encounter learning difficulties that will be analyzed in detail in Chapter 1. A solution needs to be found for improving or simulating the learning experience.

A system named ASTUS is being developed. It was conceived in ASTUS lab of the University of Sherbrooke. This system allows the student to learn knowledge in virtual labs with the help of an intelligent tutor. Several other works on this issue have been done in the past research [Kauf 03] [Tcho 02]. This project will concern knowledge representation. The motivation for developing this project is to know if a lab can be built to simulate restriction digestion and reconstructing DNA problems and if it can be integrated into ASTUS system.

ASTUS system is based on Intelligent Tutoring Systems (ITS), so before this lab begin to be designed and developed, some basic concepts about ITS will be introduced. It includes its definition, goal, architecture and content.

The architecture of traditional ITS is normally composed of four components: the Expert

Model, the Student Model, the Tutor Model, and the Learning Environment. The Expert Model contains the knowledge and the information referred to as domain or content knowledge. The Student Model can be thought of as containing an advanced database of the student's knowledge. The Tutor Model contains teaching information. The main purpose of the Tutor Model is to reduce the knowledge difference between the expert and the student to a minimum or to none. Finally, the Learning Environment is the interaction interface between students and the system.

Since the genetic lab must be integrated into ASTUS system, some specific characteristics and some distinct differences in ASTUS system will also be introduced compare to a traditional ITS. In ASTUS system, knowledge is separated from other models. Knowledge is taken as an independent part and this part stores all knowledge relative to the domain in one database. Expert Model, Student Model, and Tutor Model are taken as engines. When they need knowledge or information from the system, they can communicate with the knowledge model.

There is an assumption that if knowledge can be reified relative to problems of restricting digestion and reconstructing DNA, encoded with corresponding objects and tools and integrated into the simulation interface, the student and the tutor may benefit from this intelligent lab. The design of the interface should also satisfy students' interest in this domain and motivate them to participate in learning restriction digestion and reconstructing DNA with a computer simulation.

Some software is used as development tools for this project. PowerPoint is used to draft the interface for simulating genetic lab. XML tool is used to store information relative to knowledge and JAVA is used to create communication tools. The final interface design will use Flash.

In this project, knowledge is reified relative to problems of restricting digestion and

reconstructing DNA by using the structures of representation of knowledge based on *l'étudiant virtuel* [Maye 01].

In *l'étudiant virtuel*, frames have described each type of knowledge. Every individual frame may be seen as a data structure. The slots in the frame contain information such as:

- Frame identification information;
- Relationship of this frame relative to other frames;
- Procedure information on use of the structure described;
- Frame default information.

The data representation structure has never been used in knowledge representation domain; representing knowledge relative to restriction digestion and reconstructing DNA will evaluate and test these structures as appropriate or not. This project will also detect any deficiency in these structures. As well, the implementation will benefit the understanding of these data structure. One can get specific and detailed information relative to *l'étudiant virtuel* from Chapter 3.

In this project, all procedural and semantic knowledge relative to restriction digestion and reconstructing DNA have been identified and simulated. Semantic knowledge corresponds to primitive concepts of the laboratory or a concept built using the tools of the laboratory of conceptual experimentation and concepts already defined. Next, procedural knowledge corresponds to the execution code of an existing tool or a way of using a combination of existing tools. Episodic knowledge is also recorded into the database for the tutor agent control about the actions of the student.

By simulating and integrating restriction digestion and reconstructing DNA into ASTUS system, it is believable that student and tutor from the biology domain will benefit from this project.

Chapter 1

Restriction Digestion and Reconstructing DNA

1.1 Purpose

The specific purpose of this laboratory experiment is to help students to learn the operations of restriction digestion and DNA reconstruction and help them to master the principles of DNA reconstruction. To do this, students will cut DNA with a series of restriction enzymes, single or in combination. DNA fragments will be separated by agarose gel electrophoresis. The fragments of the digestion reaction will be used to construct a restriction map. A restriction map gives relative positions of cutting sites. The restriction map will confirm the identity and sequence of DNA [Wats 93].

1.2 Principle

Restriction digestion is the definition for digesting DNA into pieces or parts with restriction enzymes [Lodi 00]. These special enzymes recognize specific sequences in the DNA molecule wherever that sequence occurs in DNA.

It is assumed that different restriction enzymes are used and digestions are complete. Different enzymes or a combination of any of them digests clones of DNA, and different groups of fragments are obtained by groups of enzymes. DNA fragments are separated by fragment weight with using the agarose gel electrophoresis.

DNA fragments are loaded into an agarose gel slab, which is placed into a chamber filled with a conductive electrophoresis buffer. A direct electric current is passed between wire electrodes at each end of the chamber. The negatively charged DNA particles migrate towards the positively charged end of the chamber. The matrix of the agarose gel acts as a molecular sieve through which smaller DNA fragments can move more easily than larger size DNA fragments. Over a period of time smaller DNA fragments will travel farther than larger ones. DNA fragments of the same size will stay together and migrate in discrete bands. The DNA fragments cut by different types of restriction enzymes will produce a unique banding pattern [Lodi 00].

The restriction sites of DNA from these fragment groups may be tried to reconstruct. The problem can be taken as:

From:

- Fragment groups from enzyme A
- Fragment groups from enzyme B
- Fragment groups from enzyme C
- Fragment groups from enzyme A and B
- Fragment groups from enzyme A and C
- Fragment groups from enzyme B and C
- Fragment groups from enzyme A, B and C

To:

- Reconstructing DNA
- Establishing restriction map for relative positions of cutting sites
- Confirming the identity and sequence of DNA

1.3 Example

The 8 kb circular DNA of the cauliflower mosaic virus was made linear by digestion with the *BstEII* restriction enzyme and then digested with *SalI*, *XhoI*, or a combination of the two enzymes [Glic 94].

SalI and *XhoI* have their own recognition sites for cleavage. So, DNA will be cleaved by these enzymes in different fragments. These fragments can be obtained by agarose gel electrophoresis.

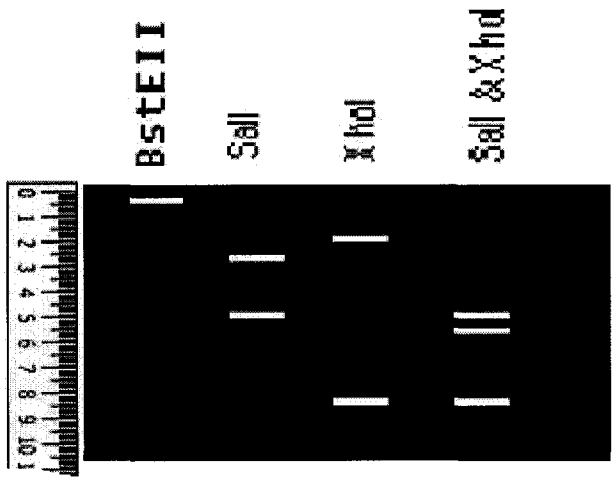

Figure 1. DNA Fragments in agarose gel electrophoresis

As Fig.1 shows, the fragments appear in the gel. Every white strip represents a fragment of DNA, the fragment sizes can be measured from agarose gel electrophoresis using the ruler (the value 0 cm aligns to the top of the gel) to get centimeter values. Enzyme *SalI* cleaves DNA in two fragments; one of the fragments is measured at 4.7 cm with the ruler.

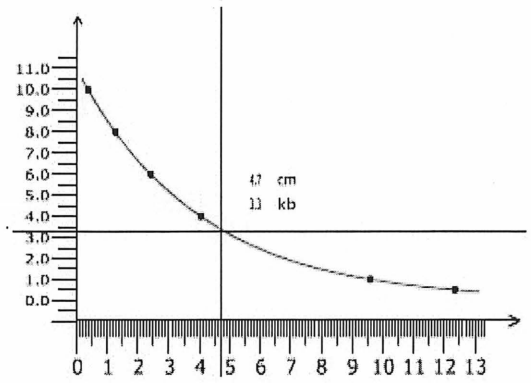

Figure 2. Curve line for changing cm to kb

As Fig.2 shows, a curve line is drawn for changing cm to kb (measure unit for DNA fragment size). Axis x is for cm and axis y takes as kb. The relationship between cm and kb is based on the equation ('A' is the consistence of agarose):

$$kb = e^{-A\,cm}$$

The size of one fragment of *SalI*, 3.3 (kb) can be retrieved from the graph at 4.7 (cm). As the result, *SalI* cleaves the DNA in two fragments; the size of every fragment is 3.3 and 4.7 (kb). *XhoI* also cleaves two pieces of fragments: 6.5 and 1.5 (kb). The combination of the two enzymes cleaves three fragments: 3.3, 3.2, and 1.5 (kb).

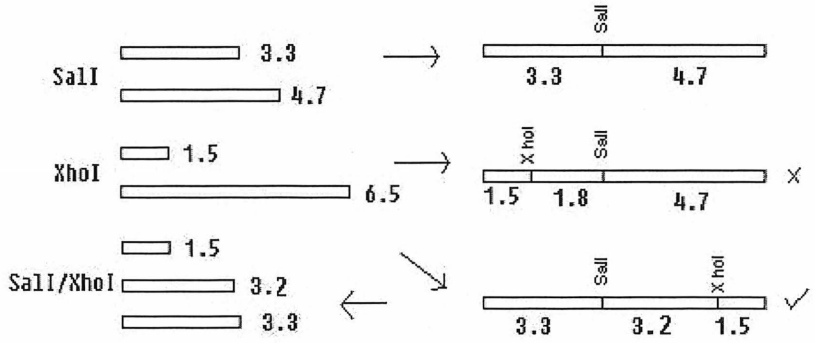

Figure 3. Principle for reconstructing DNA

As Fig.3 shows, supposing enzyme *SalI* cleaves at 3.3 kb, if enzyme *XhoI* is used for another digestion, there should be two possibilities:

1) Enzyme *XhoI* cleaves at 1.5 kb, then there are three fragments with enzyme SalI together, and the sizes of the fragments are: 1.5, 1.8 and 4.7 (kb);
2) Enzyme *XhoI* cleaves at 6.5 kb, then there are three fragments with enzyme SalI together, and the sizes of the fragments are: 3.3, 3.2 and 1.5 (kb).

Compare to the two fragments results with double digestion measured from the gel, when enzyme *SalI* cleaves at 3.3 kb, enzyme *XhoI* should cleave at 6.5 kb, not at 1.5 kb. From single and double digestion, the relative positions of restriction sites are deduced.

1.4 Problem

The problems of restriction digestion and reconstructing DNA are very complicated in the biology domain and genetic studies. It is important for students to learn about the principles of this area. Furthermore, it is also necessary for students to participate in laboratory activities to fully understand the knowledge and master the experiment operations. The skills of students will be enriched by learning how to restrict digestion and reconstruct DNA and this knowledge will be important layout for their further study [Glic 94]. Unfortunately, there are some difficulties during the lab experiment for restriction digestion and reconstructing DNA:

- Preparation of an experiment is complicated; it will take up the tutor a lot of time to prepare the utensils, tools and experiment materials.
- Restriction enzymes are expensive. Every vial of enzymes costs from $40 to $200.
- Restriction enzymes are delicate and need to be treated carefully. Since enzymes are proteins and proteins will be denatured as the temperature increases, enzymes must be kept in a freezer until they are used.

- For each digestion, it needs at least 1 hour or longer for reaction.
- For every experiment, at least 3 or 4 enzymes will be chosen to do digestion; duplicate work will increase the burden of student and tutor.
- Every step of experiment needs precise operation without any mistakes. Any mistakes may result in inaccurate data; therefore affect the reconstruction of DNA.
- Troubleshooting must be done on site by the tutor when the students meet problems during the experiments. It is difficult for the tutor to monitor operations of each student and give them corresponding instructions.

All the problems mentioned will affect the processes and results of experiments and just few experiments could be made, not allowing the student to experiment until he understands well. Some students will waste a lot of materials and time; they will lose interest for what they are doing. The tutor will also get exhausted of explaining and helping students individually. Both of them want to find an easier way to solve these problems. This is part of the motivations that this project explores for.

Chapter 2

Intelligent Tutoring System and ASTUS

As described above, objective of this project is to find a way to solve the problems of restriction digestion and DNA reconstruction experiment. This will be one of the applications of the ASTUS system in biology domain. ASTUS is specially designed as an intelligent teaching environment which is an advanced new system based on the traditional Intelligent Tutoring System (ITS).

2.1 Introduction to Intelligent Tutoring System

An Intelligent Tutoring System is based on a computer training system. It incorporates the techniques of communicating and teaching knowledge and skills to students [Weng 87].

The goal of an ITS is to provide automatic one-to-one instruction. An ITS can intelligently answer student's questions and provide individualized help which is more than training simulations [Ong 00]. Knowledge representation strategies can be employed by an ITS to model student's cognitive processes. Based on the accurate model of the student's and the expert's knowledge, an ITS designs teaching strategies to provide explanations, hints, examples, demonstrations, and provides students practices as needed in terms of content and style. Students taught by an ITS generally learn faster and decode the knowledge better than those students taught in the classroom [Ong 00].

2.1.1 Architecture of traditional ITS

The architecture of traditional ITS is normally composed of four components, as Fig.4 shows: the Expert Model, the Student Model, the Tutor Model, and the Learning Environment [Weng 87] [Elso 90] [Slee 82].

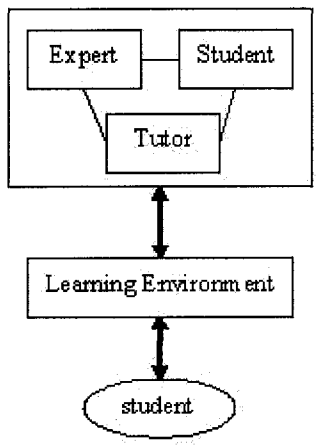

Figure 4. Architecture of traditional ITS

Expert Model

Like a human expert, the Expert Model contains the adequate knowledge and the information referred to the domain. Normally, knowledge is classified as semantic knowledge and procedural knowledge, and is maintained in a database by an expert system. Semantic database stores pieces of information about the concepts and problems in the domain, while procedural database contains knowledge of procedures and rules that an expert uses to solve problems within that domain.

Student Model

The Student Model can be taken as an advanced database of the student's knowledge. The

Student Model stores information about student's levels and abilities, such as the student's current knowledge state, that is specific to each individual student. Other general information can also be included, such as acquisition and retention of knowledge. Acquisition measures how fast students learn new knowledge and retention measures how well they recall the material over time [Elso 90]. The bandwidth of the Student Model (i.e., the quality and quantity of the input to the model) decides the accuracy and detail of this database, and determines the granularity by which the student's actions can be tracked.

Tutor Model

This component contains teaching information. Teaching strategies and essential instructions are main concepts of this model. Strategies designed must be consistent with the student's actual needs, and will lead the learning process of students naturally without the need of additional intervention by a real teacher. The ultimate purpose of this model is to reduce the knowledge differences between the expert and the student to a minimum or to none.

Learning Environment

The Learning Environment is the interaction between students and the system. The interaction is the visible aspect of the teaching and learning process. If the Expert Model is taken as an internal representation of a domain, the Learning Environment will be the external representation [Zhan 97]. Furthermore, the Learning Environment directly communicates with the student.

2.2 Role of Knowledge

As said before, knowledge runs through all processes of ITS. The Expert Model needs a good knowledge representation as support. A well-organized knowledge representation will benefit students. Tutors can compare the knowledge which the student has already learned with the

expert knowledge base, thereby pointing out the places where the student has difficulties therefore knowledge differences.

A good knowledge representation will also lead to a good solving strategy. The student will lose their directions easily when solving a problem if the required knowledge objects and the relationships among these knowledge objects are incomplete. Knowledge has two basic types: semantic knowledge and procedural knowledge which is for manipulating the semantic knowledge. Solving a problem requires complete and appropriate knowledge representation [Merr 00]. This is also the same purpose this project is looking for.

A perfect and well-organized knowledge representation can enrich ITS's ability to reason and solve problems. It can be a memory aid which provides information that can be directly perceived and used without the need of explicit interpretation and formulation.

A good, integrated and well-organized knowledge representation has rich concepts classified as semantic and procedural knowledge. In which each concept is defined by relating to many other concepts, and the relationships between concepts are explicitly declared. A good resolution for achieving a goal is highly depending on the sufficiency and strictness of knowledge representation. For every goal, a good knowledge representation system may give reasonable explanations, hints and references [Gera 97].

On the contrary, the fragmentary, imperfect knowledge generally means a poor structure of concepts. Some links are weak; some are inappropriate; and some are not existent at all. If some of the inappropriate links are extremely strong, they will lead to misconceptions. When using this kind of knowledge to solve the problem and achieve the goal, the result will be incomplete, incorrect or unanswerable. This will make problem solving process much more difficult.

A student practicing in a learning system begins with a goal that they want to realize and then will try to get relevant information from the knowledge representation to help them. However, if the knowledge representation is poor, they will have difficulties in tracking through the goal to the solution, or they will never find the solution. At certain steps, the students may get possible clues leading to the goal, but they may miss the correct or formal solutions. As the students try to apply the knowledge they gained for the next higher goal, they may find an inadequacy in their knowledge base. All of these will be the potential limitations brought by a poor knowledge representation design. The Students are generally unsuccessful in the end in these situations and lose their interest. Therefore, it is important and necessary to design a good knowledge representation for the domain [Gera 97].

2.3 Knowledge Representation in ASTUS

The ASTUS was conceived in ASTUS laboratory of Université De Sherbrooke. ASTUS integrates the concepts of Learning Environment and Intelligent Tutoring System; it allows the student to learn knowledge in virtual labs under the supervision of the intelligent tutor which works at minimizing the difference between the expert and the student.

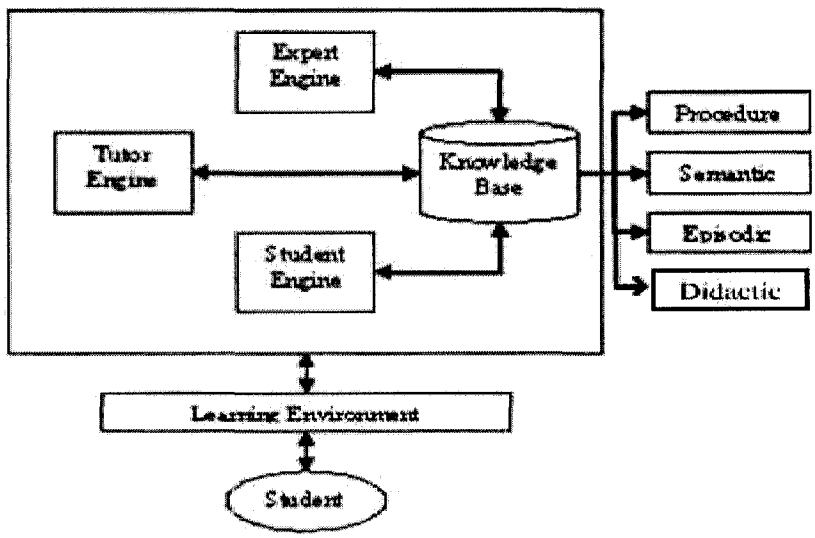

Figure 5. Architecture for ASTUS

2.3.1 Knowledge Base Component

Due to the importance of knowledge representation, as Fig.5 shows, in ASTUS system, the knowledge base is made as an individual component. Compare to traditional ITS, it stores all the domain knowledge into one database, no longer separating expert, student and tutor knowledge. Generally these knowledge can be classified as:

- Semantic knowledge: information about concepts and problems.
- Procedural knowledge: information about procedures and rules.
- Episodic knowledge: information and actions that the student has done.
- Didactic knowledge: information on how to teach semantic or procedural knowledge.

Expert Model, Student Model, and Tutor Model are now designed as engines. When they need knowledge or information, they communicate with the knowledge base model. The expert engine could access any kind of knowledge in knowledge base; it can solve any problem with the knowledge within the domain. Student engine may access only part of the content in knowledge base. The amount of knowledge accessed by the student engine is determined by the student's state and tutor strategy. At the same time, the tutor engine minimizes the gap between the student's knowledge and the expert knowledge. The tutor engine designs the corresponding strategies and instructions to help scaling up the student's knowledge body. All the information the student ever retrieved and all the actions the student has performed will be recorded as episodic knowledge. From the episodic knowledge, the tutor engine can infer what the student knows and what he is able to do. Subsequently the tutor engine modifies or updates its teaching strategies and contents. Until now, the student engine and the tutor engine are not implemented yet in ASTUS system and they will be implemented later. This project presents some solutions regarding the knowledge representation and tutor engine.

2.3.2 Displaying knowledge

In ASTUS system, conceptual objects and tools are created in a graphical interface. Analysis and teaching of knowledge are carried out easily by the student and the tutor. The visual objects on the user interface are used to represent semantic knowledge and the sequences of operations to manipulate the objects represent the procedural knowledge. Tools will execute tutorial sections and help the student when needed. The interface design for ASTUS system has a mixture of description of the learning environment and the representation of it by the student.

At the back-end of the system, all the objects are described as semantic knowledge. All procedural knowledge is composed of the execution of an existing tool or the way of using a combination of existing tools. All episodic knowledge is saved to the database for monitoring by the tutor engine. Didactic knowledge implies knowledge regarding the selection of relevant teaching and learning strategies.

Chapter 3

Encoding and Presenting Knowledge in ASTUS: Tools and Methods

3.1 Hypotheses and Assumption

The assumption of this project is based on the statement by and stipulates that if the knowledge representation employed for teaching is similar to those used by the student to encode knowledge then the training is facilitated [Ande 90] [Merr 00]. An assumption is added that:

If:

1) knowledge can be encoded in the tutoring system the same way in which the student encodes it,
2) and the interface clearly presents the concepts and the processes,

Then:

The diagnosis of the student knowledge will be facilitated.

This assumption brings out two important actions: encoding knowledge and clearly present them to the student. The purpose of this project is to validate this assumption by integrating those two actions within the framework of the ASTUS system. This project develops a virtual genetic lab in ASTUS system. The structures of representation of knowledge based on *l'étudiant virtuel* is used [Maye 01] to describe restriction digestion and DNA reconstruction problems, and reify semantic, procedural, and episodic knowledge related to restriction digestion and DNA reconstruction problem. Essentially, restriction digestion and DNA

reconstruction problem solving learning can be improved by constructing a good knowledge representation.

Knowledge related to restriction digestion and DNA reconstruction problems is decoded and integrated into the graphical interface and proves that the virtual lab will be a better way of training students. It is possible to simulate DNA digestion with enzymes and to obtain the photographs of the gel corresponding to these digestions in ASTUS system. The visual simulation of experimentation allows the students to observe the phenomena as if in the real world. It is also possible to reify concepts of DNA, enzymes, and fragments of DNA related to restriction digestion and DNA reconstruction. The fragments of DNA are not visible during real experimentation but they can be conceptually represented by sticks. The students will be able to use the tool which looks like the ruler of the laboratory in real experimentation to identify the size of these fragments.

Because the goal of this project is to let the student learn how to reconstruct DNA according to the corresponding references and rules, the students already have some knowledge about the genetic domain (i.e. they know the principle of DNA digestion with enzymes). So, the design of the Learning Environment is suited for their knowledge state and purpose in this domain, and motivates them to participate in learning restriction digestion and DNA reconstruction with a computer simulation.

3.2 Software tools used in developing genetic lab

In the project, several softwares are used: PowerPoint as draft and presentation tool; XML (eXtensible Markup Language) as knowledge representation tool; Java as communication tool between the interface and the knowledge; Flash as interface design tools.

3.2.1 Draft Design with PowerPoint

At the beginning of this project, PowerPoint is firstly used to draft the interface of the genetic lab. PowerPoint is a complete presentation graphics package. It provides all the tools and functions Ones needs to produce a professional-looking presentation.

The interface has been conceived in two steps. Firstly a draft has been drawn by using PowerPoint. Then it has been validated by an expert in ITS filed, André Mayers and a professor in biology, Claude Déry. Therefore, the interface has to fulfill the following points:

- Every student should have his own account to log in.
- After the students log in, they should be able to resume their previous studies done during their last experiment or practice session.
- The student cannot select DNA by himself. The tutor will have the exclusive right to decide which DNA the student should have. The student can only make a choice between the types of DNA: linear or circle. For example, if the student chooses linear, the system will automatically assign DNA *ldna2* to this student.
- Each experiment in the lab allows having 10 enzymes in maximum.
- The student cannot directly measure DNA fragment in kb (which is specially used to measure DNA); he must draw a curve line representing the relationships between cm and kb. After that, he can transform DNA fragment size from cm to kb.
- Depending on the knowledge state of the student, the tutor may decide which functions the student can use in the genetic lab. For example, if the result of DNA fragment measuring is wrong, the tutor will not allow the student to continue to the next step.
- The states of current experiment in the virtual lab may be saved if the student wants to have a pause or wants to resume his work next time.
- If a student finds that his DNA reconstruction result is wrong, he can roll back his reconstruction process by deleting the bad cut points and then adding in new ones.
- For multiple digestions, only double and triple digestions can be accepted.

- If the student feels some digestion results are not satisfied, the student can practice the experiment again, choosing that digestion another time and re-measure those fragments.

All these suggestions founded the basis for modifying the expressions and correcting the functions in the interface design. After several times of drafting, presenting, discussing and suggesting, a blue print for the interface of the genetic lab comes out. The final interface design is shown in Chapter 4.

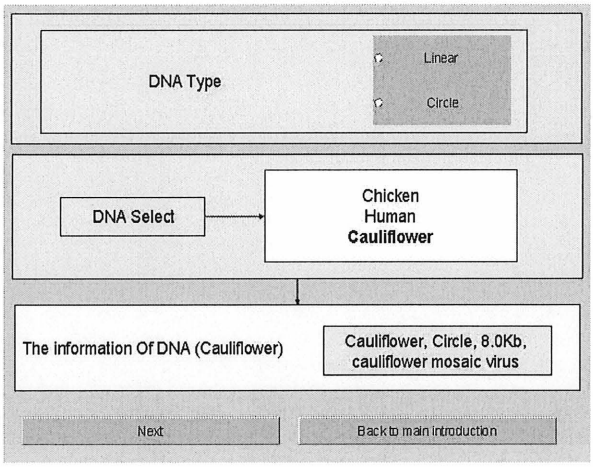

Figure 6. Draft for DNA select

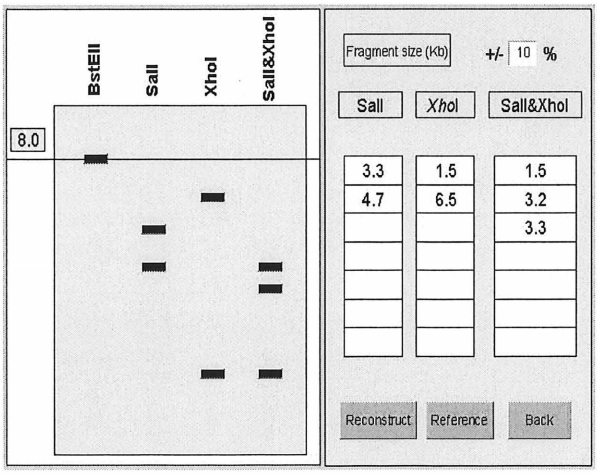

Figure 7. Draft for digestion DNA

Fig.6 and Fig.7 show many changes compared to the final interface design (see also Fig.12-15 in Chapter 4). If building the virtual genetic lab directly in the interface, and creating the functions based on previous understanding, any minor changes on interface may cause a lot of troubles in the background: removing components, redrawing pictures and re-coding functions.

Using PowerPoint is really a good method for drafting the interface at the first step, since it is easy to modify and edit draft interfaces. The time and work loads for designing and modifying the interface of the genetic lab was greatly decreased.

3.2.2 Knowledge Representation with XML

This project uses XML to store the data of knowledge base model. XML is an industry-standard, system-independent way to represent data [Arms 02].

There are some reasons for using XML to represent knowledge:

- Plain Text: Since XML file is similar to standard text file which can be created and edited with any tools: from a standard text editor to a visual development environment. This point makes it possible and easy when a knowledge base need to be modified and updated according to the level and request of student and tutor.
- Data Identification: XML represents the Meta information of data, instead of storing displaying format. It may identify knowledge by putting the information of knowledge attribute in XML tags.
- Data Extraction: Regular and consistent tags make it easier to build a program to extract XML data. The XML file must be well-formed tree structure. For example, the <dna> tag must always have a </dna> terminator, otherwise the XML parser won't be able to read the data. On the other hand, as XML is a vendor-neutral

standard, there are several XML parsers to choose from, any of which can reduce my work from processing XML data.

- Hierarchical: Finally, XML documents benefit from their hierarchical structure. Hierarchical document structures are, in general, faster to access because it can be drilled down to the part needed using appropriate search strategies.

3.2.3 Communication with Java

Java issued as a tool for developing programs connecting between different components of the system. Since Java is an object-oriented language, it can build the model of knowledge easily [Ecke 02]. Concepts in semantic knowledge are modeled as Java classes. Procedural knowledge is modeled as Java methods or constructor of the Java class which represent the corresponding semantic knowledge the methods will operate on. In addition, some controller classes are implemented responsible for communication between knowledge classes and Flash interface. The reasons of choosing Java to construct the communication component between XML and Flash are:

- As flash can only read information from XML files but is restricted from writing data into XML files at present, another tool must be chosen between flash and XML for inter-connection and writing data into XML file which is for the knowledge base.
- Java is the best tool for connecting between flash and XML because of its strong bonds with web components and XML. Java has already established a powerful, complete parser package for accessing and extracting the data in XML files.
- Another benefit brought by java and XML is: by using XSLT (XML Stylesheet Language for Transformations) it is easy to generate Java code directly from XML files when needed.
- For generality, the interface is separated from the program logics. With flash and java program, it is much easier to update and modify the functions of project without affecting much on the flash interface.

3.2.4 Interface design with Flash

The final interface design will apply Flash as the developing tool. Flash is a powerful tool for designing animated simulations. Comparing to other software, it has several obvious advantages:

- Speed of loading: It's a vector graphics based system (images are not stored in pixels, but in vectors). The file made in flash is pretty small and can be loaded fast.
- Easy to integrate: Any web browser can support embedding flash by simply installing the flash player component.
- Easy to create simulations: Flash supports creating vivid and good-looking effects like onmouseover buttons and animated graphics with different kinds of elements (movies, buttons, etc.). It provides strong and sufficient functions for simulating a real lab.
- Simplicity of communication: it is simple to communicate between Flash and Java. The necessary information can be collected from Flash and pass as parameters to the back-end java objects for further process. Also, Java methods are used to pass the processed data to Flash for displaying.

3.3 Theory basis of knowledge representation

As discussed above, knowledge representation is very important for ITS and ASTUS system. So representing and encoding knowledge will be the emphasis of this project. The document *l'étudiant virtuel* is referred [Maye 01] as the theoretical basis of this project. This document is named *l'étudiant virtuel* as it is focusing on the structures of representation of the knowledge used by the student and the expert. It adopts frame based system for modeling a knowledge domain.

The document *l'étudiant virtuel* was written by André Mayers and is under complete. Some implementations have been done as a proof of concept of this document. According to the document *l'étudiant virtuel*, a knowledge base combines the following elements: semantic knowledge, procedural knowledge and episodic knowledge.

3.3.1 Semantic Knowledge

In order to describe information about concepts and problems, semantic knowledge is used. Semantic knowledge can be classified as following [Maye 01]:

- Goal: Intention or objective to be reached or achieved.
- Function: Allows referring to a concept without naming it explicitly.
- Relation: A logical or natural association between two or more objects.
- Other: The other concept that is not a function, relation, or goal.

Example for classifying semantic knowledge

In a genetic lab, Java.lang.String is a primitive concept and is also an 'other' concept that is not function, relation, or goal; GoalDigestDNA is a primitive concept, and at the same time, it is also a goal.

Content of semantic knowledge

In semantic knowledge, each concept has a frame and there are a certain number of basic slots for describing its contents in this frame. The concept may have other additional slots if it is a goal, a function or a relation.

a) The concept is not a goal, a function or a relation.

Every concept has a unique identifier being used to identify this concept. First of all, when describing the content of a concept, the goal slot is a special slot which stores a list of all the possible goals available to be achieved by this concept. For example, concept DNA can realize the goal GoalDigestDNA with DigestDNA procedure. Secondly, if this concept has some concepts which it inherits from or which inherit from it, they should be recorded as super concepts or sub concepts in the content of this concept. Thirdly, in the content of the concept, a list of identifiers of procedures is also included which are relating with this concept. For instance, concept DNA has three procedures: GetDNA, CloneDNA and DigestDNA. Finally, there is other information in the content of the concept such as follows:

- Name: A name is given to the concept which could be used by the system and be presented by the user.
- Description: A definition of the concept short and precise in plain text.
- Creator: The creator of the concept.
- Pedagogic: The didactic resources associated with this concept with indices on their use according to the context.

b) The concept is a goal.

If the concept is a goal, except for the above slots mentioned it must include the ability description that represents the goal's function, the parameters which are used to realize this goal and the procedures which the student could use to achieve this goal. For instance, the procedure CloneDNA is related to the goal GoalCloneDNA and it will use the Concept DNA. For a goal, it also has didactic strategy. Didactic strategy is used for tutor engine to teach how to achieve this goal.

c) The concept is a function or a relation

If the concept is a function or a relation, it can have other additional slots except the above slots mentioned such as parameter, association, goal of function and so on. In this project, function and relation are not implemented.

3.3.2 Procedural Knowledge

Types of Procedural Knowledge

Procedural knowledge has two types:

1) Primitive Procedures: Corresponds to a goal and realization of that goal.
2) Complex Procedures: Composed of several goals that can be realized with one procedure or more.

Content of Procedural Knowledge

Every procedure has a frame and there are a certain number of basic slots for describing its contents in the frame. A procedure has a unique identifier being used to identify this procedure within the system. This procedure can realize a goal, and this goal is described by goal identifiers together with any number of identifiers of concepts which are used to achieve this goal. For example, procedure CloneDNA can realize the goal GoalCloneDNA with the concept DNA.

The content of the procedure should contain one of the following four values to verify the procedure validity:

- Valid: this procedure makes it possible to achieve the goal.
- False: this procedure does not make it possible to achieve the goal.
- Valid and not always finished: A valid procedure, but which is not always finished.

- False and not always finished: An invalid procedure, but which is not always finished.

Every procedure corresponds to a method or a constructor of a class for processing concepts. As Fig.8 shows, if the procedure is an ordinary method, it should have a method name, an instance of class name and some parameters. If the procedure is a constructor, it should have a constructor name and parameters.

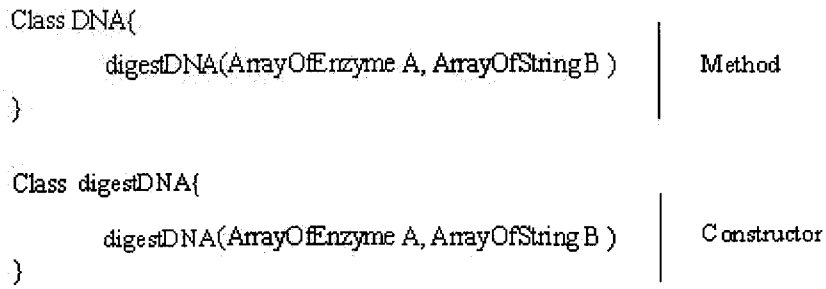

Figure 8. Method and Constructor correspond to the procedure

There is other information in the content of the procedure such as follows:

- Script: describes what the procedure carries out.
- Context: makes it possible to define other constraints to use a procedure.
- Utilization: contains preference (useful, useless, and probable) which makes it possible for agents to judge the relevance of the procedure according to the context. These preferences become crucial factors when there are several valid procedures competing to achieve the goal.
- Diagnostic solution: The member diagnostic-solution contains two parts: teaching strategy and preference. The preferences make it possible to identify a possible difficulty of the student while the application of the teaching strategy helps the student to overcome this difficulty.

In the content of the procedure, it also contains a list of references pointing to resources, examples, exercises and tests.

3.3.3 Episodic Knowledge

There exists another kind of knowledge: episodic knowledge. Generally, episodic knowledge doesn't belong to the knowledge representation of a domain, since it consists of records of student's actions in a virtual laboratory. Episodic knowledge will allow us to replicate and redo the student's steps, to check if the student knows how to reach a goal, to know the student's weaknesses and strengths, and to give him some personalized explanations or help based on the results of analysis.

There are two types of episodic knowledge: episode and cognition.

Episode

An episode is a trace of the actions of a student when he or she tries to accomplish a goal. Each episode corresponds to a goal. If the goal has sub goals, the episode will have sub episodes.

Every episode has also a unique identity. It has a time slot to record when the episode was done. Every episode will record the way of realizing a goal by recording the sequences of identities of the cognitions which are used for resolving this goal. In the content of the episode, it contains the procedure that the episode used. If the episode has super episode or sub episode, their identities should be recorded in the content of this episode. Every episode must also have a slot to record some result cognitions which is obtained by this episode. For evaluating the episode state, the result for cognition of an episode should be one of these states: success, failed, waiting. At last, the cost of the episode should be known. Usually a cost of 1 is associated to every episode accomplished by a non-complex procedure. Episodes accomplished by a complex procedure have a cost that is the sums of all of its sub episodes.

Cognition

Cognition is an instance of a concept used in a specific episode. Every cognition has a unique identifier being used to identify this cognition. It has the identity of the episode in which this cognition occurs. If this cognition is a component of other cognitions, the identities of the other cognitions should be recorded which including current cognition. Specially, if the cognition is an instance of a described concept, the identities of the sub cognitions of the described concept should be recorded down. If the cognition is an instance of primitive concept, the value such as Integer should be put, Boolean in the content of cognition. If this cognition was previously used in another episode, the identity of the episode should be recorded in which it was used last time.

Chapter 4

Results

The structures of knowledge representation from *l'étudiant virtuel* have been presented in Chapter 4 and the principles of restriction digestion and reconstructing DNA are introduced in Chapter 1. In this Chapter, the structures of knowledge representation will be used to describe the problem of restriction digestion and reconstructing DNA. We will emphasize on decoding knowledge relative to a real lab, implementing it in a simulated genetic lab and then integrating it into ASTUS system.

4.1 Knowledge Representation of a Genetic Laboratory

All knowledge is formatted and stored in a XML file and transformed to Java classes or methods for implementation in the simulation.

4.1.1 Semantic Knowledge in a Genetic Lab

Fig.9 shows all semantic knowledge related to a genetic lab where digestion restriction and DNA reconstruction take place. In a genetic lab, Semantic knowledge includes two types of concepts: the 'other' concept which is not a function, relation or goal, and the second concept which is a goal. These concepts, with procedural knowledge, will realize the corresponding goals. For example, the concept DNA, when it is used to execute the DigestDNA procedure, will achieve the goal GoalDigestDNA, and get GroupOfDNAFragment concept.

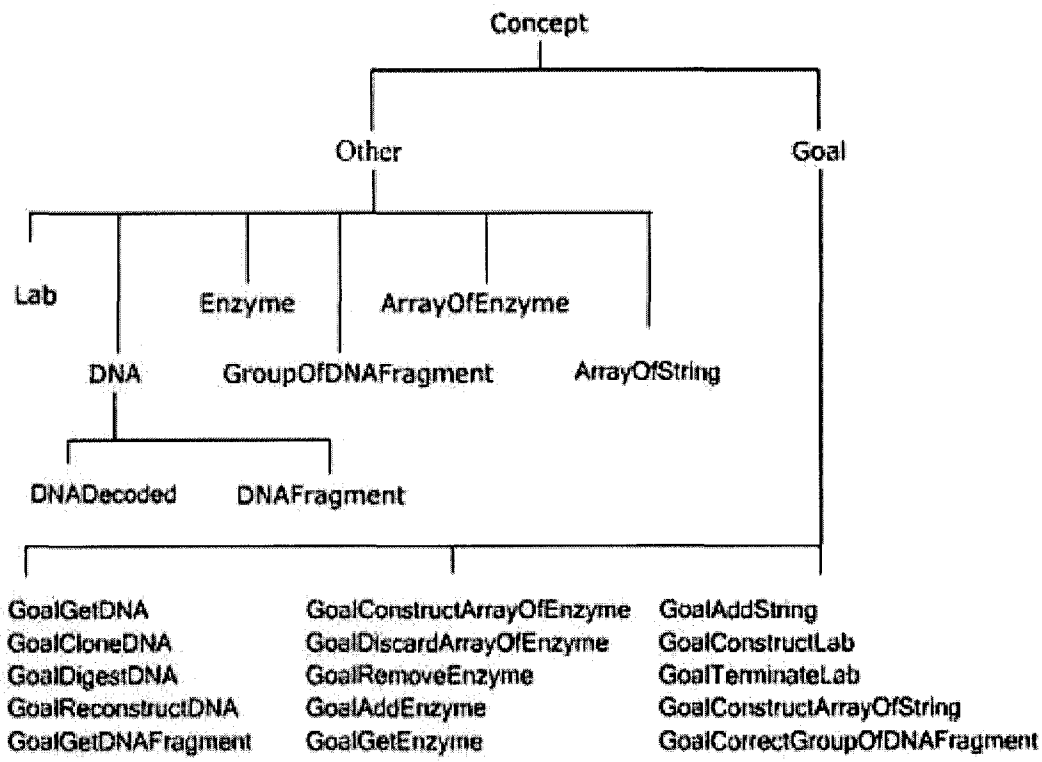

Figure 9. Semantic Concept Structure

Other concept:

Here are all 'other' (not a goal, a function or a relation) concepts in genetic lab:

- **Lab**: it is a concept which controls all the system.
- **DNA**: it records information about DNA, including name, type, length and origin.
- **DNADecoded**: it records information about DNA which has been decoded by some enzymes.
- **DNAFragment**: it is part of DNA.
- **Enzyme**: it records information about Enzyme including name, recognition site and origin.

- **GroupOfDNAFragment**: it is a group of DNA fragments from digestion. These fragments' length can be obtained by student's measuring.
- **ArrayOfEnzyme**: single enzyme or combination of enzymes for digestion.
- **ArrayOfString**: group of fragments' size values.

All concepts will be recorded and described according to the structure of knowledge representation. When the structure of knowledge representation is used for describing genetic knowledge, some solutions are made to solve some empty or unfinished concepts:

- If the slot is not implemented, "not implemented" will be put.
- If there is no content in the slot, "Null" will be put.

For example, DNA is described in Tab.1.

Table 1. Concept DNA

Frame	**Content**
id	geneticlab.other.DNA
name	DNA
description	The concept of the DNA
creator	Xin
type concept	Primitive concept
goal	• (GoalCloneDNA,geneticlab.other.DNA) • (GoalDigestDNA, geneticlab.other.DNA, geneticlab.other.ArrayOfEnzyme, geneticlab.other.ArrayOfString) • (GoalReconstructDNA, geneticlab.other.DNA, geneticlab.other.ArrayOfString,

	geneticlab.other.ArrayOfString)
super concept	• Java.lang.Object
sub concept	• geneticlab.other.DNAFragment • geneticlab.other.DNADecoded
constructor	• GetDNA • CloneDNA • DigestDNA
parts	see below the table
pedagogic	not implemented

Concept DNA has a unique id geneticlab.other.DNA to identify this concept. Another way to classify this concept is that it is a primitive concept. The concept may realize the goal GoalCloneDNA, GoalDigestDNA and GoalConstructDNA with the procedures: GetDNA, CloneDNA and DigestDNA. Concept DNA inherits from Java.lang.Object, whereas Concept DNAFragment and concept DNADecoded inherit from concept DNA.

Here is something that should be specifically mentioned. Since the information relative to concepts DNA and Enzyme is more complex, their attributes should be put into the slot "parts" of the frame such as DNAName, DNALength and so on. In this project, an indirection is used to record all their attributes as a dependent part connecting to a corresponding concept. For instance, some main attributes related with concept DNA are put in another table, as Tab.2 shown in.

Table 2. Extension for concept DNA

Frame	**Content**
id	dna1

name	dna1
type	linear
length	9.5
origin	ASTUS

In episodic memory, when the student use DNA concept, only the id of concept DNA was recorded as dna1. The information related to dna1 can be obtained in extension frame by its id.

To apply the concept of DNA, information will be stored in a XML file that looks like this:

```
<concept>
    <id>geneticlab.other.DNA</id>
    <name>The concept of DNA<name>
    <description>The concept of the DNA</description>
    <creator>XIN</creator>
    <type concept>concept primitive</type concept>
    <goal>
        <prototype_goal>
            <idgoal>GoalCloneDNA</idgoal>
            <idconcept>geneticlab.other.DNA</idconcept>
        </prototype_goal>
        ...
    </goal>
    <super concept>
        <idconcept>Java.lang.Object</idconcept>
    </super concept>
    <sub concept>
        <idconcept>geneticlab.other.DNAFragment</idconcept>
        <idconcept>geneticlab.other.DNADecoded</idconcept>
    </sub concept>
    <constructor>
```

```
            <idproc>GetDNA</idproc>
            <idproc>CloneDNA</idproc>
            <idproc>DigestDNA</idproc>
        </constructor>
        <parts/>
        <pedagogic />
    </concept>
```

For every 'other' concept, Java class is built for communicating between Flash and XML as follows:

```
class DNA extends Java.lang.Object {
            private String dnaName;
            private String dnaType;
            private double dnaLength;
            private String dnaOrigin;

    public DNA (String name, String type, double length, String origin) {
            dnaName = name;
            dnaType = type;
            dnaLength = length;
            dnaOrigin = origin;
    }

    getName() {}
    setName() {}
    ......

    cloneDNA() {}
    digestDNA() {}
    reconstructDNA() {}
}
```

Every field corresponds to an attribute of the DNA concept, and every attribute has getter and

setter methods to retrieve and modify the attributes of the DNA class. Every method corresponds to procedural knowledge for this lab.

Goal

Here are all the goals in genetic lab:

- **GoalGetDNA**: The goal is for getting DNA with GetDNA procedure.
- **GoalCloneDNA**: The goal is for getting duplicate DNA with CloneDNA procedure.
- **GoalDigestDNA**: The goal is for getting GroupOfDNAFragment with DigestDNA procedure.
- **GoalReconstructDNA**: The goal is for getting DNADecoded with ReconstructDNA procedure.
- **GoalGetDNAFragment**: The goal is for getting DNAFragment with GetDNAFragment procedure.
- **GoalConstructArrayOfEnzyme**: The goal is for getting an empty container of enzymes with ConstructArrayOfEnzyme procedure.
- **GoalDiscardArrayOfEnzyme**: The goal is for discarding the container of enzymes with DiscardArrayOfEnzyme procedure.
- **GoalRemoveEnzyme**: The goal is for discarding enzyme with RemoveEnzyme procedure.
- **GoalAddEnzyme**: The goal is for adding enzyme to the enzyme container with AddEnzyme procedure.
- **GoalGetEnzyme**: The goal is for getting Enzyme with GetEnzyme procedure.
- **GoalConstructArrayOfString**: The goal is for getting an array of string which holds the sizes of DNAFragments in GroupOfDNAFragment with ConstructArrayOfString procedure.
- **GoalAddString**: The goal is for adding size of DNAFragment to the ArrayOfString with AddString procedure.

- **GoalConstructLab**: The goal is for constructing the Lab with ConstructLab procedure.
- **GoalTerminateLab**: The goal is for terminating the Lab with TerminateLab procedure.
- **GoalCorrectGroupOfDNAFragment**: The goal is for correctingGroupOfDNAFragment with CorrectGroupOfDNAFragment procedure.

The concept contains the basic slots such as id, name and additional content recorded. For example, the goal GoalDigestDNA is described in Tab.3:

Table 3. Goal GoalDigestDNA

Frame	**Content**
ID	GoalDigestDNA
name	The goal to digest DNA.
description	The goal is to get DNA fragment and reconstruct it.
creator	Xin
type concept	Primitive concept
goal	Null
super concept	geneticlab.Goal
sub concept	Null
constructor	Null
Parts	Null
Pedagogic	not implemented
ability	Digest DNA.
parameter	• geneticlab.other.DNA • geneticlab.other.ArrayOfEnzyme

	• geneticlab.other.ArrayOfString
procedures	DigestDNA
didactics strategies	not implemented

Concept GoalDigestDNA is a goal; it has its own id, name, ability and other slots. GoalDigestDNA will be achieved by the procedure DigestDNA with the concepts: DNA, ArrayOfEnzyme and ArrayOfString. This project doesn't implement pedagogic and didactic strategies. The goal GoalDigestDNA is not a class, since it does not have any attributes.

4.1.2 Procedural Knowledge in a Genetic Lab

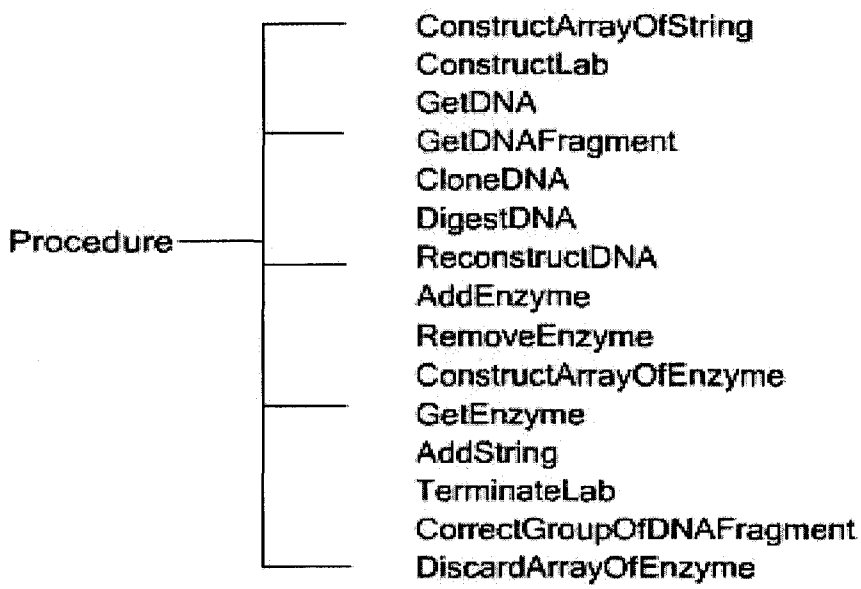

Figure 10. Procedural Knowledge Structure

Fig.10 shows all the procedures related to a genetic lab about digestion restriction and DNA reconstruction problems. Every procedure or the combination of procedures with semantic conceptual knowledge will realize the corresponding goals. For example, the procedure

DigestDNA with the semantic concepts DNA, ArrayOfEnzyme and ArrayOfString will realize the goal GoalDigestDNA, and get another concept, GroupOfDNAFragment.

Here are all the procedures:

- **ConstructLab**: the lab will be constructed.
- **GetDNA**: concept DNA will be loaded into the lab. For every lab, only one DNA can be accepted.
- **GetEnzyme**: concept Enzyme will be loaded into the lab. The lab can contain more than one enzyme, which will be selected and put into ArrayOfEnzyme for single or multiple digest DNA using AddEnzyme procedure.
- **AddEnzyme**: add Enzyme to ArrayOfEnzyme.
- **RemoveEnzyme**: remove Enzyme from ArrayOfEnzyme.
- **DiscardArrayOfEnzyme**: this procedure will discard an ArrayOfEnzyme
- **DigestDNA**: it uses Enzyme (s) (single or multiple) in ArrayOfEnzyme to digest DNA, and the student will measure all the fragments in the gel, the goal GoalDigestDNA will be realized and get a new instance of GroupOfDNAFragment. For each ArrayOfEnzyme, it must execute DigestDNA one time, so DNA will be cloned every time by CloneDNA procedure.
- **CloneDNA**: clones DNA for digestion.
- **CorrectGroupOfDNAFragment**: every GroupOfDNAFragment can be modified because the error may happen when measuring DNAFragment.
- **GetDNAFragment**: it will get the DNAFragment according to the size.
- **ReconstructDNA**: it will reconstruct DNA, get DNADecoded which inherit from DNA and some characters have been decoded by used Enzyme.
- **TerminateLab**: Lab will be terminated.
- **ConstructArrayOfString**: construct an empty container for String
- **ConstructArrayOfEnzyme**: construct an empty container for Enzyme.

Every procedure will be recorded and described according to the structure of knowledge representation mentioned before. For example, the structure of procedure DigestDNA is described in Tab.4:

Table 4. Procedure DigestDNA

Frame	**Content**
Id	DigestDNA
creator	Xin
goal	(GoalDigestDNA, geneticlab.other.DNA, geneticlab.other.ArrayOfEnzyme, geneticlab.other.ArrayOfString)
arguments	• (geneticlab.other.DNA, dna) • (geneticlab.other.ArrayOfEnzyme, arrayofenzyme) • (geneticlab.other.ArrayOfString, arrayofstring)
script	not implemented
validate	valid
context	true
method	typemethod: method methodJava: digestDNA instance: geneticlab.other.DNA nameparameter: • geneticlab.other.ArrayOfEnzyme • geneticlab.other.ArrayOfString
notes	Not implemented for utilization, diagnostic solution, resource didactics, example, exercises and test.

As Tab.4 shows, procedure DigestDNA has a unique id "DigestDNA" used to identify this procedure. This procedure can realize the goal GoalDigestDNA with the concepts DNA,

ArrayOfEnzyme and ArrayOfString, and get one instance of concept GroupOfDNAFragment. The procedure DigestDNA is done successfully. Other slots are not implemented in this project such as script, context and so on. This procedure is only evaluated by verifying the DNAFragment. For DigestDNA procedure, it corresponds to one method of class DNA as follows:

```
GroupOfDNAFragment digestDNA
        (ArrayOfEnzyme arrayofenzyme, ArrayOfString arrayofstring )
{
        return new GroupOfDNAFragment ();
}
```

The procedure information will be stored in a XML file that looks as follows:

```
<procedure>
    <id>DigestDNA</id>
    < creator >XIN</creator >
    <goal>
        <prototype_goal>
            <idgoal>GoalDigestDNA</idgoal>
            <idconcept>geneticlab.other.DNA</idconcept>
            <idconcept>geneticlab.other.ArrayOfEnzyme</idconcept>
        </prototype_goal>
    </goal>
    <arguments>
        <param>
            <idconcept>geneticlab.other.DNA</idconcept>
            <nameparam>dna</nameparam>
        </param>
        <param>
            <idconcept>geneticlab.other.ArrayOfEnzyme</idconcept>
            <nameparam>arrayofenzyme</nameparam>
        </param>
```

```
        <param>
            <idconcept>geneticlab.other.ArrayOfString</idconcept>
            <nameparam>arrayofstring</nameparam>
        </param>
    </arguments>
    <script/>
    <validate>valide</validate>
    <context/>
    <utilisation/>
    <method>
        <typemethod>method</typemethod>
        <methodJava>digestDNA</methodJava>
        <instance>dna</instance>
        <nameparam>arrayofenzyme</nameparam>
        <nameparam>arrayofstring</nameparam>
    </method>
    < diagnostic_solution />
    < resource_didactics />
    <examples/>
    <exercises/>
    <test/>
</procedure>
```

4.1.3 Episodic Knowledge in genetic lab

In ASTUS system, cognitions of episodic knowledge are also recorded and described according to the structure of knowledge representation introduced before. One example of the cognition in genetic lab is described in Tab.5.

Table 5. A cognition of episode

Frame	**Content**
id	1068835308358

id episode	1068835308357
concept id	geneticlab.other.DNA
subparts	Null
super parts	Null

As Tab.5 shows, the cognition was recorded at time 1068835308357 when the student used the concept DNA, and the id of cognition is 1068835308358. When the student triggers an action, which corresponds to one or more procedures, the episode will automatically record what has occurred at that time.

Every episode of episodic knowledge will be recorded and described according to the structure of knowledge representation mentioned before. An example of an episode is described in Tab.6.

Table 6. An episode

Id	1068835308406
Time	1069196997403
Goal	(GoalDigestDNA, 1068835308408, 1068835308409, 1068835308410)
Procedure	DigestDNA
Result	1068835308407
State	success
super episode	Null
sub episode	Null
Cost	1

As Tab.6 shows, this episode was recorded at time 1069196997403 and its id is

1068835308406. It used procedure DigestDNA, realized the goal GoalDigestDNA, and got the result 1068835308407 which corresponds to concept GroupOfDNAFragment. The state of this episode was marked success.

4.2 Knowledge implementation and integration

The knowledge representation has been implemented and integrated into the interface design of the genetic lab. All procedural knowledge and semantic knowledge related to restriction digestion and DNA reconstruction have been identified and simulated. All semantic knowledge of the DNA domain is composed of primitive concepts in the laboratory and composite concepts which are built on defined concepts by using the tools of laboratory during virtual DNA reconstruction experimentations. Procedural knowledge corresponds to the execution code of an existing tool or a way of using a combination of existing tools. Episodic knowledge is also recorded into the database.

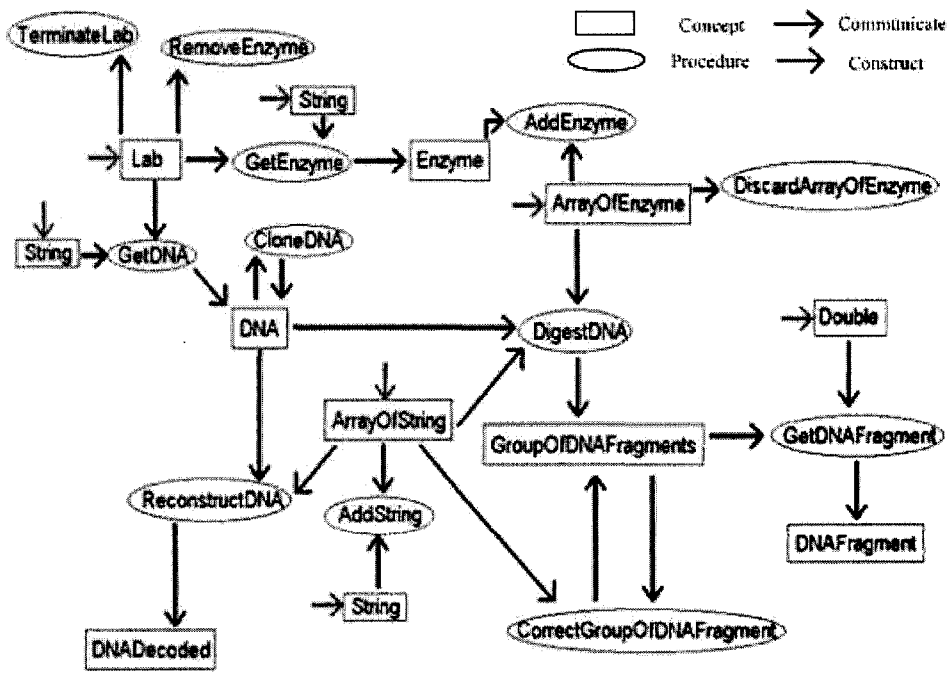

Figure 11. Knowledge Structure in genetic lab

Fig.11 shows the knowledge representation in a genetic lab. The steps can be followed to see every episode with its cognitions.

4.3 The Learning Interface

Now, it will be explained how the knowledge representation is implemented and integrated into a learning interface which integrates a virtual lab for digestion DNA and reconstructing DNA in the genetic lab:

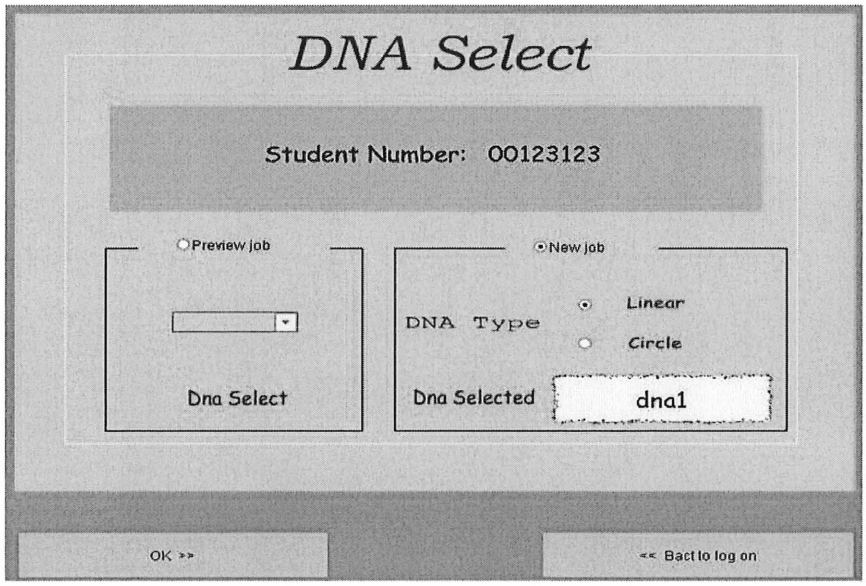

Figure 12. Interface for DNA select

As Fig.12 shows, when the student logs in to the virtual genetic lab, the concept Lab is automatically constructed by the procedure ConstructLab and realizes the goal GoalConstructLab. DNA is the first concept that the student must choose for the genetic lab. For every student, the lab will assign one DNA, which was defined by the tutor. There are two types of jobs: the previous job stores data that the student has done before and the new job lets the student begin a new lab. When the student clicks the "Ok" button, the procedure GetDNA will be executed with the concept DNA that has been selected and the goal

GoalGetDNA is obtained.

As Fig.13 shows, for restriction digestion and DNA reconstruction, the student must select enzymes. All enzymes will be listed on the left listbox, when the student clicks the button ">", one enzyme will be selected and shown on the right listbox. To achieve these steps, three procedures are automatically executed and three goals are correspondingly obtained: GetEnzyme with GoalGetEnzyme; ConstructArrayOfEnzyme with GoalConstructArrayOfEnzyme; and AddEnzyme with GoalAddEnzyme. Button "<" accompanies two procedures and corresponding goals: DiscardArrayOfEnzyme with GoalDiscardArrayOfEnzyme and RemoveEnzyme with GoalRemoveEnzyme.

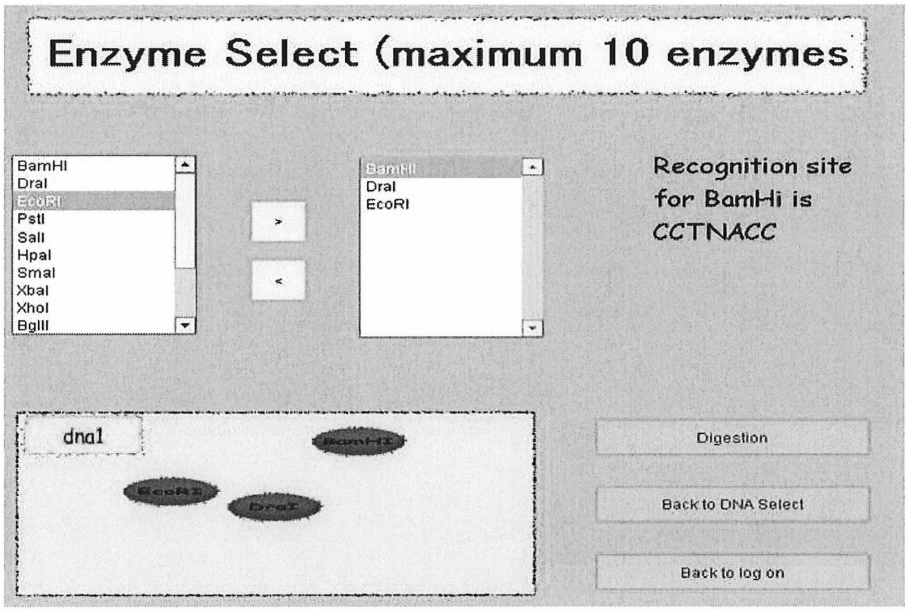

Figure 13. Interface for Enzyme select

When both DNA and enzymes are selected, the lab will simulate a real lab to do DNA digestion with the given enzymes. For every ArrayOfEnzymes, when the student clicks button "Digestion", there is a DNA cloned by the procedure "CloneDNA". The lab will automatically do CloneDNA procedure, get the goal GoalCloneDNA and generate the simulating gel for the purpose of measuring DNAFragment.

As Fig.14 shows, the result of digestion will be shown on the interface, which looks like a real gel map. The student will measure every DNAFragment and input the value on the right. When the student has finished measuring all the DNAFragments, button "Make map" which corresponds to the procedure "DigestDNA" and the goal GoalDigestDNA will lead the student to next step.

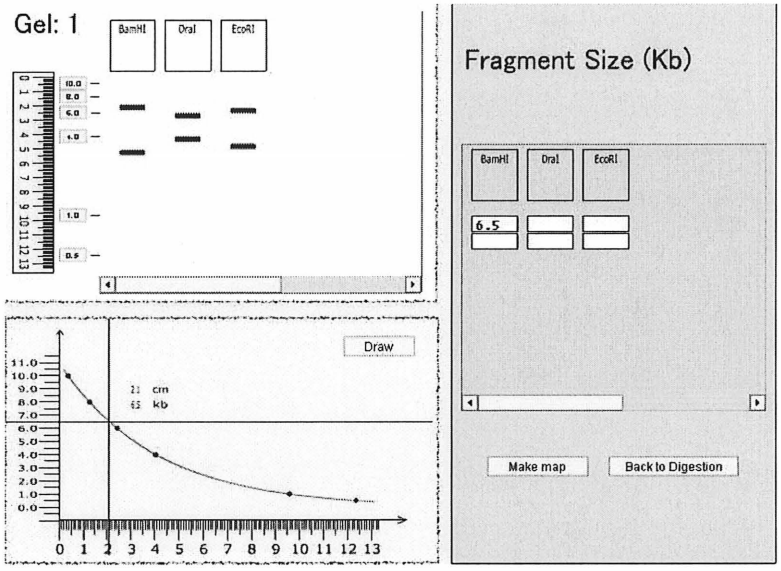

Figure 14. Interface for digestion DNA

As Fig.15 shows, all references taken by the student from the gel are listed at the top. Each group of references corresponds to every GroupOfDNAFragment which are obtained by procedure DigestDNA. From the cutting tool's combo box, the enzyme can be selected and cut point value can be input into the text field on right side. When the student clicks button "Make cut" which corresponds to the procedure ReconstructDNA, the cut clip will appear at the place specified on the DNA sample. This will result in a DNADecoded concept and realize the goal GoalReconstructDNA.

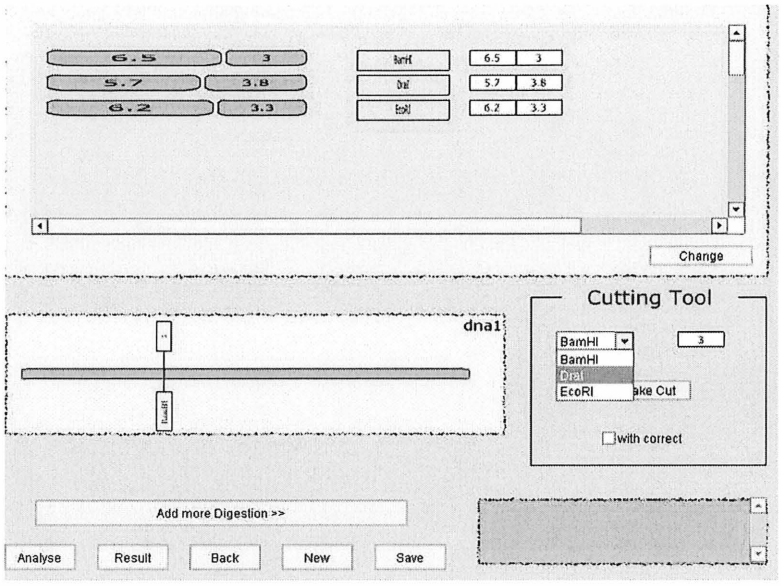

Figure 15. Interface for reconstructing DNA

Other semantic concepts and procedural knowledge were also implemented by some buttons such as: button "New" which corresponds to the procedure ConstructLab and the goal GoalConstructLab; button "Change" which corresponds to the procedure CorrectGroupOfDNAFragment and the goal GoalCorrectGroupOfDNAFragment. When the lab is closed, it corresponds to the procedure TeminateLab and the goal GoalTeminateLab.

As a kind of simulation in a genetic lab, some objects are built to help the student handle the concepts related to the knowledge. Such as:

- Simulate a real digestion container as a graphic area, and when adding or removing enzymes, they will automatically show on the interface.
- Design the photograph of the gel corresponding to the real digestion gel. The student will be able to use the simulated ruler to identify the size of DNA fragments.
- Draw the curve line which looks exactly like the student did on the paper to calculate kb value from cm.
- Use a clip which recording enzyme's name and cut point on DNA to simulate that the

student reconstructs DNA in the real lab. These clips can be erased easily which corresponding to a student's correcting action of the DNA reconstruction.

All these simulations have improved the feasibility of execution of real experiments and allowed the student and the tutor agent to observe the phenomena as if it happened in the real lab.

4.4 Tutoring Process in ASTUS

Since in ASTUS system, for every lab, an intelligent tutor should always be in presence, some control functions have been built for the tutor agent. The function of the tutor agent is to monitor the student's actions. If the student does not follow the procedure pre-outlined, the agent must validate the answer of the student at each step, and if it is proved to be false, the tutor agent can stop the process and show the feedback to the student, including the correct methods and other helpful hints.

Here are some tutor agent controls in the genetic lab:

- Control of enzyme choice: a maximum of 10 different enzymes is allowed.
- Control of DNA fragment measures: if the size of the fragment which was measured by the student goes beyond the acceptable bias of the correct value, the system won't let the student go to the next step for DNA reconstruction.
- Control of DNA reconstruction: if the reconstruction result doesn't satisfy some reference, when the student validates the result, the system will show the student which reference is not matching with the result.

By integrating and simulating restriction digestion and reconstructing DNA into ASTUS system, it is believable that there will be the following benefits:

- The tutor can predetermine the experiment once, and use it as a model forever.
- Using simulation materials instead of real materials will save money.
- Simulating reactions may save the operation time and decrease probabilities of error.
- Simulation lessens the burden of the real tutor.

Conclusion

We believe that the comprehension of a scientific domain may be substantially improved by finding a good knowledge representation. A perfect and well-organized knowledge representation enhances the ability to reason and solve problems.

Knowledge has been described for restriction digestion and reconstruction of DNA molecules with enzymes in a genetic lab based on a document named *l'étudiant virtuel.* Semantic and procedural knowledge related to a genetic lab have been encoded in the tutorial system the same way in which the student encodes it; the interface clearly presents the concepts and the processes. The student's learning process will be facilitated by the graphical interface and the tutor agent will manipulate the episodic knowledge from the student more easily.

This genetic lab has been integrated into ASTUS system which is based on intelligent tutoring system. It will be put into practice in the near future.

It is believable that knowledge representation for a genetic lab will contribute to relevant works. Also we still have further research and development to do in the future. For example, some procedural knowledge are automatically done by the system, they do not generate perfect and complete episodes really done by the student. It would be preferable if some actions could be designed to simulate these procedures and have the student execute the tasks. On the other hand, ASTUS system needs to implement the student engine and the tutor engine. Intelligent tutor agent control still needs some improvements such as catching the process of measuring DNA fragment rather than validating the final result, giving the possible reconstruction result in terms of the given references. The structure of *l'étudiant virtuel* also needs to be modified and improved.

References

[Ande 90] Anderson, J. R. (1990). *The adaptative character of thought.* Lawrence Erlbaum and associates.

[Arms 02] Armstrong, E. (2002). *The Java Web Services Tutorial.* Addison-Wesley Pub Co.

[Cohe 82] Cohen, P. R. & Feigenbaum, E. A. (1982). *The Handbook of Artificial Intelligence: 3.* William Kaufmann, Inc.

[Davi 93] Davis, R., Shrobe, H. & Szolovits, P. (1993). What is a Knowledge Representation? *AI Magazine*, 14(1):17-33.

[Ecke 02] Eckel, B. (2002). *Thinking in Java, 3.* Prentice-Hall.

[Elso 90] Elsom-Cook, M. (1990). Guided Discovery Tutoring. In Elsom-Cook, M. (Ed.) *Guided Discovery Tutoring for ICAI research.* London: Paul Chapman Publishing. 3–23.

[Fire 88] Firebaugh, M. W. (1989). *Artificial Intelligence: A Knowledge-Based Approach.* PWS-Kent, Massachusetts.

[Gera 97] Gerace, W. J., Dufresne, R. J., & Leonard, W. J. (1997). A framework for the storage of knowledge and its implications for problem solving. *UMPERG technical report,* 1-19.

[Glic 94] Glick, B. R. & Pasternak, J. J. (1994). *Molecular Biotechnology, Principles and Applications of Recombinant DNA.* ASM Press.

[Jong 91] de Jong, T. (1991). Learning and instruction with computer simulations. *Education & Computing, 6*: 217-229.

[Kauf 03] Meudja, B. K. (2003). Conception et implémentation d'une couche de communication entre les agents intelligents et les laboratoires virtuels. Master thesis in University of Sherbrooke.

[Lodi 00] Lodish, H. & Berk, A. (1995). *Molecular Cell Biology, 4.* W H Freeman & Co.

[Luge 02] Luger, G. F. (2002). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving,4.* London: Addison-Wesley.

[Maye 01] Mayers, A. (2001). L'étudiant Virtuel. Unpublished.

[Merc 90] Mercer, L., Prusinkiewicz, P. & Hanan, J. (1990). The concept and design of a Virtual Laboratory. In *Graphics Interface '90 Conference proceedings.* Canadian Information Processing Society. 149-155.

[Merr 00] Merrill, M. D. (2000). Knowledge objects and mental models. In D. A. Wiley (Ed.). *The Instructional Use of Learning Objects.* Bloomington: Association for Educational Communications and Technology.

[Ong 00] Ong, J. & Ramachandran, S. (2000). Intelligent Tutoring Systems: The What and

The How. *Learning Circuits*, 1(2).

[Paqu 98] Paquette G. (1998). Meta-knowledge representation, application to learning systems engineering. *TL-NCE technical reports*.

[Paqu 99] Paquette, G., Aubin, C. & Crevier, F. M. (1999). A Knowledge-based Method for the Engineering of Learning Systems. *Journal of Courseware Engineering*, 2.

[Slee 82] Sleeman, D. and Brown, J. S. (1982). *Intelligent Tutoring Systems*. New York: Academic Press.

[Tcho 02] Tchoumtchoua, J. (2002). Conception et implémentation d'un serveur de modèle de l'usager. Master thesis in University of sherbrooke.

[Wats 93] Watson, J. D. & Gilman, M. (1992). *Recombinant DNA, 2*. W H Freeman & Co.

[Weng 87] Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann Publishers, Inc.

[Zhan 97] J. Zhang. The nature of external representations in problem solving. *Cognitive Science*, 21(2):179-217.