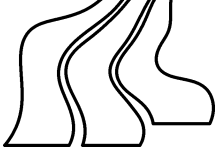
AALBORG UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
FREDRIK BAJERS VEJ 7 - DK 9220 AALBORG - DENMARK



Title:

Computerized Adaptive Test based on Bayesian Network for basic operations with fractions

Project period:
2001.09.01 – 2001.12.19

Project group:
d313a

Members:
Linas Būtėnas
Agnė Brilingaitė
Alminas Čivilis
Yin Xuepeng
Nora Zokaitė

Advisor:
Jiři Vomlel

Abstract:
In this work we present the Computerized Adaptive Test (CAT) for basic operations with fractions. The motivation is the aim to apply the theory of the Bayesian Networks to a practical problem. The main results of our work are the detailed analysis of statistical results from paper tests, a modeled network and an implemented CAT program.

Report 1

Pages: 49

Copies: 8

Preface

December 19, 2001, Aalborg University

This project is written by E1-116 on the DAT3 semester 2001, under the theme "Computerized Adaptive Test based on Bayesian Network for basic operations with fractions".

The main purpose of the project is to design the probabilistic network and to implement the CAT program.

We wish to thank Kirsten Bangsø Jensen, a teacher of the secondary school in Brønderslev.

Alminas Čivilis

Agnė Brilingaitė

Linas Būtėnas

Xuepeng Yin

Nora Zokaitė

Contents

Introduction

This project was done as a practical part for the course of Decision Support Systems at Aalborg University. The main goal was to get knowledge about Bayesian network theory and to apply it to a practical problem.

Since the theory of Bayesian networks has become a greatly developed branch of Artificial Intelligence research it is used in building models of CAT as well[9]. A CAT is a computerised adaptive test that can cover any teaching domain. As we have a sufficient mathematical background for basic algebraic operations we chose the domain of fractions for our CAT.

Our goals were:

- to get acquainted with BN theory,
- to analyse the domain of fractions,
- to build a student model and an evidence model based on Bayesian networks for fractions domain,
- to implement the CAT, and
- to verify the behavior of the implemented CAT.

The first chapter of our report introduces the necessary part of theory of Bayesian networks. The rules of creation of a student and evidence models are described in details and the definitions (for Bayesian network, entropy, best item selection) are given. The features of the CAT in comparison with paper tests are presented.

The second chapter concerns the analysis of fractions and the practical process of building a student model. The operations with fractions are analysed. The structure of a student model is given. The process of network learning is described. The analysis of the paper tests is presented.

The CAT implementation is described in the third chapter. The architecture of the CAT is presented. Hugin Decision Engine is introduced as a tool for the Bayesian network management.

Chapter 1

Theoretical background for CAT

In this chapter we discuss about Computerized Adaptive Test (CAT) and its differences from the paper-based tests. The basic theory for modeling CAT described in following sections.

1.1 Computerized Adaptive Tests compared with the paper tests

Computer technology has given new opportunities for the educational and psychological testing. Computerized Adaptive Test optimizes the administration of the tests and adds new features, which were impossible in the paper tests. Here are the CAT features:

- it is available all year around, so it is possible to reduce the number of students that takes the test at one time (in one place);
- it is available in many locations (if it is presented through the Internet or some network);
- it is better tailored to the ability level of each student;
- scores appear on a screen immediately following the test;
- it provides more precise information about the student abilities than the normal paper test;
- number of asked questions is reduced, so it saves testing time;
- it decreases examinee frustration since low-ability examinees are not forced to take test items constructed for high-ability testees, and vise versa;
- each result could be recorded and easily reused later for various purposes (for example: to calculate the result for the whole class, school or the group of students; to upgrade the CATs primary believes about the knowledge of some particular student group making the test more precise);
- it improves the variability of test, because the test contains an entire item bank, rather than merely 20 or 40 specific items that make up the examinee's test. As a result, it is more difficult to answer the test by summarizing all the answers to the items or the

sequence of right answers. However, the item bank must be quite large to ensure that test items do not repeat frequently;

- it can include text, graphics, photographs, audio records and even full-motion video clips;
- it does not require the examinee to be a computer specialist when answering a test. He or she can make it without previous computer experience. The test requires only basic computer skills and if the test is quite important, usually there is a tutorial which gets you acquainted with the basic computer skills and the nature of the CAT test before beginning the official test.
- it is possible to compare the CAT results of the students, because all the test questions are rated before students takes test. Then computer counts the score from the correctly and incorrectly answered questions, it takes the difficulty level into consideration as well.

We have listed above the advantages of the Computerized Adaptive Test. There is one feature which can be hardly evaluated whether is it positive or negative. The main idea of CAT is that "the computer scores each question before selecting the next one. You must answer each question when it is presented. For this reason, once you answer a question and move on to another, you cannot go back and change your answer. The computer has already incorporated both - your answer and the requirements of the test design into its selection of your next question". So you can not change your previous[1] answers if you realize you have answered wrongly, and you can not use the next questions to help you to answer the one you have at the mean time. Looking from one point of view it is good - you solve the item using your own knowledge. From the other point of view if the current item seems difficult to you or you know that you knew how to solve it 5 minutes ago and you have forgotten it now you can not leave it and return to it later. As you can see it is quite a tricky feature of the CAT.

What are the disadvantages?

- To test a class of students you need to have a computer for every student if you want them to take a test at the same time.
- It is harder to concentrate for the test siting by a computer. When you have a paper test, you realize that it is something important.
- Some people are "afraid of computers" and get nervous because they think they will be unable to manage a computer test

Bergstron and Gershon [5] had described the main CAT features in one sentence: "When the difficulty of items is targeted to the ability of candidates, maximum information is obtained from each item for each candidate, so test length can be shortened without loss of reliability."

1.2 Building a Student model.

The goal of our project was to construct a computerized adaptive test of students' skills we wish to measure using an item bank. The first step in the construction of the CAT was to create Bayesian Network (build in Hugin Proffesional 5.7 [3]) for a Student model (SM) and Evidence models ($EMds$) [9].

Definition 1.2.1 [7] A Bayesian network consists of

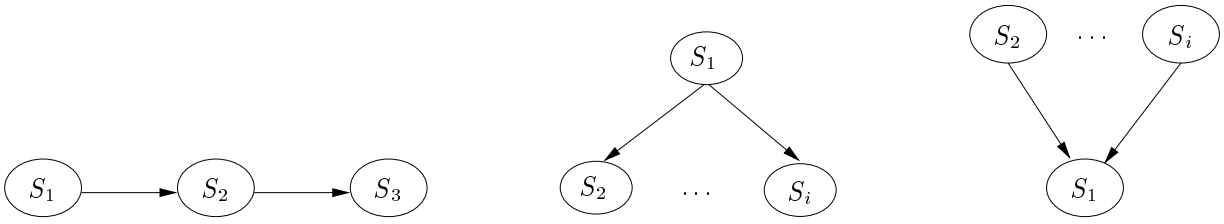
- a set of variables and a set of directed edges among variables;
- each variable has a finite set of mutually exclusive states;
- the variables together with the directed edges form a directed acyclic graph (DAG);
- to each variable A with parents $B_1, \ldots, B_n$, a potential table $P(A|B_1, \ldots, B_n)$ is attached.

Let $S_1, \ldots, S_N \in \{0, 1\}$ be a set of discrete random variables and each

$$S_i = \begin{cases} 0 & \text{with probability } p_{S_i 0} \\ 1 & \text{with probability } p_{S_i 1} \end{cases}, \quad p_{S_i 0}, p_{S_i 1} \in [0, 1], \quad p_{S_i 0} + p_{S_i 1} = 1, \quad i \in [1, N].$$

Student model (SM) is a Bayesian network which describes relations among student's skills measured by random variables $\mathbf{S} = (S_i)_{i \in [1,N]}$ and a probability distribution $P(\mathbf{S}) = P(S_1, \ldots, S_N)$ represents knowledge about a student.

SM was build combining these types of connections:



(a) Serial connection

(b) Diverging connection

(c) Converging connection

Figure 1.1: Types of connections

These are the three basic types of connections among directly connected variables S_1, S_2 $\ldots S_i$. SM was build according to the rule of $d-separation$:

Definition 1.2.2 [7] Two distinct variables S_1 and S_2 are $d-separated$ if, for all paths between S_1 and S_2, there is an intermediate variable V such that either

- the connection is serial or diverging and V is instantiated

  or

- the connection is converging, and neither V nor any of V's descendants have received evidence.

Following this rule, we could indicate whether any pair of skill variables were independent given the evidence 1.3
In case of a converging connection in our *SM* we can define different types of relations. For example one of the parent variables $S_2 \ldots S_i$ can have a greater impact on S_1 than other parents or all the parents $S_2 \ldots S_i$ can have an equal impact on S_1.

Special kinds of relations are logical - logical *or* and logical *and* connections. These connections can be encoded into conditional probability tables as shown below:

Logical	S_1 = True		S_1 = False	
and	S_2 = T	S_2 = F	S_2 = T	S_2 = F
S_3 = T	1	0	0	0
S_3 = F	0	1	1	1

Logical	S_1 = True		S_1 = False	
or	S_2 = T	S_2 = F	S_2 = T	S_2 = F
S_3 = T	1	1	1	0
S_3 = F	0	0	0	1

Table 1.1: Logical relations of S_3 to S_1, S_2 described by $P(S_3|S_1, S_2)$

1.3 Building Evidence models

Evidence models (*EMds*) describe relations among skill variables and an item from item bank. Example:

Figure 1.2: Evidence model

here S_i denotes i- th skill variable and $X_{i,l}$ denotes i- th group of items and l-th item in the group i. A collection of items (tasks, exercises) grouped according to skill variables is called item bank. Each item has several possible answers; $5-6$ in our *EMds*. Some of the answers correspond to misconceptions (described in Section 2.4.3), one of the answers is correct.

In the *EMd* each item $X_{i,l}$ can only have one skill variable as a parent. In case it would have two or more parents we created an intermediate node. In each *EMd* the guessing probability is encoded in the conditional probability tables $P(X_i|S_i)$. Example:

	S_1 = True	S_1 = False
X_1 = T	0.95	0.2
X_1 = F	0.05	0.8

Img	S_1 = True	S_1 = False
	0.75	0.25

Table 1.2: 'Imagination' task given imagination skill S_1

Having imagination skill examinee will solve the task with probability 0.95. Without this skill the probability of the correct answer is not 0, but equals to 0.2 because of the guessing probability from 5 possible answers.

Evidence on a variable is an information about the state of the variable. Evidence can only be transmitted to the *SM* through an *EMds*.

1.4 EM algorithm

In this section we will shortly describe the problem of maximum likelihood estimation of the parameters having incomplete data [8]. The Expectation-Maximization (EM) algorithm iteratively computes the maximum-likelihood estimates from the incomplete data. Each algorithm iteration consists of two steps - expectation (E) step followed by maximization (M) step. Incomplete data could be understood like two sample spaces $\mathcal{X}$ and $\mathcal{Y}$ with "many to one" mapping from $\mathcal{X}$ to $\mathcal{Y}$. The observed data y is from space $\mathcal{Y}$. The data x is observed indirectly through y, and is lying in $X(y)$. $X(y)$ subset is computed using equation $y = g(x)$. It could be seen more clearly from Figure 1.3.

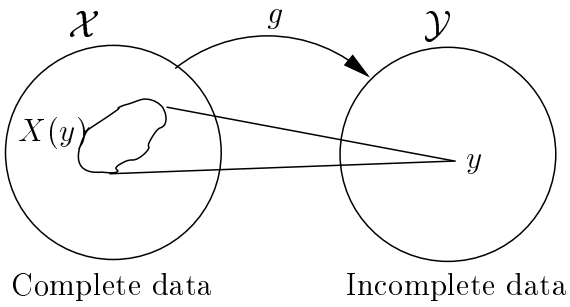


Figure 1.3: Incomplete Data model

Assume $f(x|\theta)$ is the PDF (probability density function), for the complete data, where θ is a vector of all probability tables in the Bayesian network.
Given the initial estimate of θ the EM-algorithm iterates the following two steps, until the algorithm converges:

- In the E-step the algorithm computes the current expected value of log-likelihood for all possible θ'.
$$Q(\theta'|\theta) = E_\theta\{\log f(x|\theta')|g(x) = y\}$$
- In the M-step the algorithm chooses θ' from the set of all θ' which maximizes $Q(\theta'|\theta)$

The main idea of the application of the EM algorithm to the learning of a Bayesian network is that if we knew the values of all the nodes in the network, the M-step (learning) would be easy. Because we have only partial data, we need to compute the expected values of all the nodes, and suppose that now all of them are observed ones. Let P be the initial estimate of the joint probability distribution of our Bayesian network. Then in the E-step we will do a calculation of expected marginal count of x_a given all y that may be filled as x_a:

$$n^*(x_a) = E_P\{n(x_a)|g(x_a) = y\}$$

$a \in A, A = \{fa(v) : v \in V\}$. $fa(v)$ is the family of the v node.

There are N independent cases $X^1, X^2, ..., X^N$, y^ν is incompleate observation on X_a^ν. To define $X^\nu(x_a)$ assume y^i is incomplete observation on X^i). $y^1, ..., y^N$. Then

$$X^\nu(x_a) = \begin{cases} 1 & \text{if } y_a^\nu = x_a \\ 0 & otherwise \end{cases}$$

Assume we have observed $y^1, ..., y^N$.

$$n^*(x_a) = E_P\Big\{\sum_{\nu=1}^{N} X^\nu(x_a|y^1, ..., y^N)\Big\} = \sum_{\nu=1}^{N} E_P\Big\{X^\nu(x_a|y^\nu)\Big\} = \sum_{\nu=1}^{N} P(x_a|y^\nu)$$

Let T be a conditional junction tree. Every node in the tree has a conditional probability table (all the tables correspond to the distribution P). After the accumulation the resulting clique probability tables over all cases ($n^*(x_a)$) we will get the estimated counts.
The M-step uses these counts to construct a new distribution P', with density p'

$$Np'(x_a) = n^*(x_a), a \in A, x_a \in \mathcal{X}_\dashv$$

$$p'(x_v|x_{pa(v)}) = \frac{n^*(x_{fa(v)})}{n^*(x_{pa(v)})}$$

The computed conditional probabilities are used for the next estimation of the complete-data counts.

1.5 Proof for Soft Evidence Update

Why use Soft Evidential Update

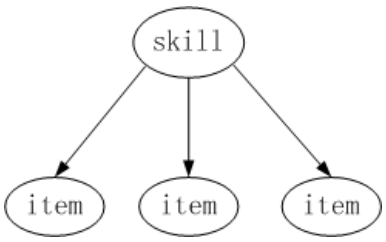


Figure 1.4: A general network

Originally, we built the evidence model, part of which looked like Figure 1.4. All the item nodes were attached to the student model. The children nodes are of the same type, and have the same conditional probabilities. During the test, some of the items would be selected to test the examinee and would never be used again but still connected to the parent node. This would make the network too big and slow down the inference. There is another way how to do the update. We attached only one node to each of the skill nodes. Every time after an item node had been used, we attached the same item node to replace the used one. By doing this, we have to edit the network and recompile it. This would again take a lot of time and system resources.

To avoid this, we proposed a better approach. The item node in Figure 1.2 is associated with an item group, but it actually represents all single items in a group, where all group members are of the same type and have the same conditional probabilities. If an item has been answered, instead of selecting a certain state of the item node, we update the network by entering the corresponding likelihoods into its parent node. This way of updating is called "soft evidential update". This will give us the same result as what we can get by selecting a state of the item node.

Introduction to Evidential Update

Evidence is a collection of findings on variables. A finding may be hard or soft. A hard finding specifies which value a variable is in. A soft finding or likelihood specifies the probability distribution of a variable. Hard evidence is a collection of hard findings. Soft evidence is a collection of soft findings. In Hugin, there are also two approaches to update the network for discrete chance nodes. An item of evidence can have the form of a statement that a variable is in a certain state, or, there is a more general item of evidence, called "likelihood". The way to do the update by selecting a certain state of some node is called hard evidence update, while the other way by entering the likelihood is named soft evidence update.

Update procedure

In Figure 1.5 the big circle represents a subnetwork of Bayesian Network which is composed of the nodes A_1,A_2,...,A_n,B. Node C stands for a node which state will be selected. The number of the nodes n could be any integer. Node B can have output or input arrows to any node in the subnetwork. Node C is connected to the network via node B. There can be any edges between nodes A_1,A_2,...,A_n,B unless they construct a directed acyclic graph. When an evidence of C being c is given, we update the network starting in node B following the next two steps

- Select the c state of node C and propagate
- Read $P(B \mid C = c)$ in node B
- Compute the likelihood of B as $L(B) = \dfrac{P(B \mid C = c)}{P(B)}$
- Retract the finding of C
- Enter the likelihood into B and propagate

Then the update is finished. The probabilities of all nodes of this network are updated.

Purpose of the proof

Since we only adopted the method described above, it is important for us to prove that the update by either selecting a state of a node or entering likelihoods to its parent has actually the same effects. Next we prove that whatever the approach is, the rest of the network is updated equivalently.

Proof for the equivalence

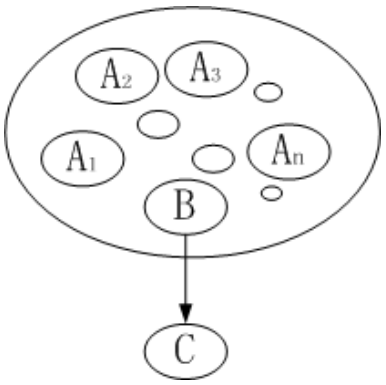


Figure 1.5: A general network

The joint probability of the network in Figure 1.5 is

$$\begin{aligned} P(A_1, A_2, \ldots, A_n, B, C) &= P(A_1, A_2, \ldots, A_n, B) \cdot P(C \mid B) \quad (1.1) \\ &= \frac{P(A_1, A_2, \ldots, A_n, B) \cdot P(C, B)}{P(B)} \\ &= P(A_1, A_2, \ldots, A_n, B) \cdot \frac{P(B \mid C) \cdot P(C)}{P(B)} \end{aligned}$$

Then we observe the evidence of C being in state c, the probability of the subnetwork could be computed as

$$\begin{aligned} P(A_1, A_2, \ldots, A_n, B \mid C = c) &= \frac{P(A_1, A_2, \ldots, A_n, B, C = c)}{P(C = c)} \quad (1.2) \\ &= \frac{P(A_1, A_2, \ldots, A_n, B) \cdot \frac{P(B \mid C = c) \cdot P(C = c)}{P(B)}}{P(C = c)} \quad (1.3) \\ &= P(A_1, A_2, \ldots, A_n, B) \cdot \frac{P(B \mid C = c)}{P(B)} \\ &= P(A_1, A_2, \ldots, A_n, B) \cdot L(B) \quad (1.4) \end{aligned}$$

Where equation 1.3 follows from 1.2 due to equation 1.1. The equation 1.4 proves that the conditional probability of the subnetwork given $C = c$ can also be computed as the joint probability of the variables in the subnetwork times the likelihood of B.

Implementation of the program

The function which implements the soft evidential update is called "updateNetwork". To update the network, we remember the beliefs of all the relative skill nodes before and after the hard evidential update from their child. Before we select a state of a child node, the function stores the beliefs and likelihoods of its parent node. After the hard evidential update, we record the new beliefs again, and calculate the new likelihood for each node. Then, we undo the update, and insert the new likelihood into the parents of the item node and do the propagation.

```
find the answered item;

get the set of its parents;

while(the parent set is not empty )
{
    record beliefs and old likelihoods;
}
select the state correspondent to the answer of the item;

propagate;

while(the parent set is not empty )
{
    record new beliefs ;
    compute new likelihoods for current parent node;
}
retract finding of the item node; propagate;

while(the parent set is not empty )
{
    insert the computed likelihoods into current parent node;
}
```

Table 1.3: pseudo-code for soft evidential update in function NetworkControl.updateNetwork

The pseudo-code of this part looks like:
The likelihoods are computed using the beliefs before and after the propagation and previously entered likelihoods, for example, a skill node B has a child node C which has m-number of states, assume we compute the likelihood of B, $L_n(B)$, where n stands for number of questions from the item group of B that were asked, while C has the evidence of being $c_k, k \in \{1, 2, \ldots, m\}$. $L_n(B)$ is the expected new likelihood, $L_{n-1}(C)$ is the previously entered likelihood. We compute $L_n(B)$ by

$$L_n(B) = L_{n-1}(B) \cdot \frac{P_n(B \mid C = c_k)}{P_{n-1}(B \mid C = c_{k-1})}$$

from the pseudo-code listed above.
Because

$$L_1(B) = L_0(B) \cdot \frac{P_1(B \mid C = c_1)}{P(B)}$$

, where $P(B)$ is the initial probability of B, and $L_0(B)$ is the initial likelihood of B which is

always 1. If we follow in the same way, then we can get

$$\begin{aligned} L_2(B) &= L_1(B) \cdot \frac{P_2(B \mid C = c_2)}{P_1(B \mid C = c_1)} \\ &= \frac{P_1(B \mid C = c_1)}{P(B)} \cdot \frac{P_2(B \mid C = c_2)}{P_1(B \mid C = c_1)} \\ &= \frac{P_2(B \mid C = c_2)}{P(B)} \\ L_3(B) &= L_2(B) \cdot \frac{P_3(B \mid C = c_3)}{P_2(B \mid C = c_2)} \\ &= \frac{P_3(B \mid C = c_3)}{P(B)} \\ &\dots \end{aligned}$$

Finally, the likelihood of B for the n-th time can be computed as

$$L_n(B) = \frac{P_n(B \mid C = c_k)}{P(B)}$$

the denominator is always the initial probability of B. By inserting the likelihoods and propagating, the network is updated, which is the way how soft evidential update works.

1.6 Next Best Item Selection

In order to build the most informative test which would contain the least number of tasks per student we used information function - *entropy*.

Definition 1.6.1 [10] Entropy $H(P)$ of probability distribution $P(\mathbf{S})$ is defined as

$$H(P(\mathbf{S})) = -\sum_{s_1,\dots,s_N \in \{0,1\}} P(S_1 = s_1, \dots, S_N = s_N) \cdot \log(P(S_1 = s_1, \dots, S_N = s_N)).$$

In our CAT this information function was used :

- as the criteria for stopping the test, i.e. after the certain value of entropy is reached - test stops;
- in the algorithm of the next item selection.

In our *EMds* each item has 5 or 6 possible answers and item bank consists of 138 items. Best next item selection algorithm:

Definition 1.6.2 Let $\mathbf{X} = \{X_1, \dots, X_M\}$ ($M = 138$ in our *EMd*) be an item bank, $\mathbf{x}^{(k-1)} = (x_1, x_2, \dots x_{k-1})$ be the responses to previously answered items $\mathbf{X}^{(k-1)} = \{X_1, \dots, X_{k-1}\}$ and $\{1, 2, 3, 4, 5, 6\}$ be a set of possible responses to any item from $\mathbf{X}$. The k-th item X_k of a myopiccally optimal test is defined to be

$$\min_{X_k \in \mathbf{X} \setminus \mathbf{X}^{(k-1)}} \sum_{x_k \in \{1,2,3,4,5,6\}} P(X_k = x_k \mid \mathbf{x}^{(k-1)}) \cdot H(P(\mathbf{S} \mid X_k = x_k, \mathbf{x}^{(k-1)})).$$

Chapter 2

Basic operations with Fractions

This chapter describes the development of the student model for fractions. The domain of fractions is introduced in the first section. Types of tasks for basic operations with fractions and the ways to solve them are analysed in the second section. The structure of our student model for fractions is described in the third section. The analysis of paper tests we gave to students is presented in the fourth section.The process of model adaptation and the results of its verification are presented in the fifth section.

2.1 Introduction to the domain

Our group has decided to analyse basic algebraic operations with fractions. The students of a secondary school solve different exercises testing their understanding of fractions and their ability to operate with them. We created a CAT that could help teachers to find out the skills students have and to diagnose the misconceptions of students. Our CAT can also be used by children themselves to clear up which skills they miss. We assume that children are able to add, subtract, multiply and divide natural numbers. We will test the skills related just to fractions. There are sometimes a few ways to solve every type of the exercises. Before constructing the Bayesian network we had to do an analysis of the domain of fractions (Section 2.4.2). We tried to imagine how students solve exercises and which operations are difficult to learn. During the analysis we have distinguished groups of skills and we prepared the items of different levels according to them. We did not analyse division in our model.

2.2 Analysis of the domain

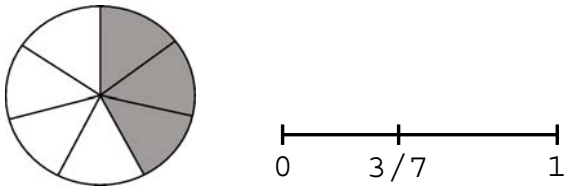
This section deals with the algebraic operations with fractions. The possible tasks with fractions are described. The necessary skills to solve the tasks are presented.

2.2.1 Fraction Imagination and Understanding

What is fraction imagination?

This is the first thing that students learn at school about fractions. The idea of fractions is often presented using the pie graph. But there are more ways to imagine what a fraction is:

one can think of it as a part of the x-axis. (Figure 2.1)



(a) (b)

Figure 2.1: Fraction imagination

Fraction imagination and understanding in our Student model

We included the skill *Img* to our Student model.

2.2.2 Comparison

Types of tasks of comparison

There are a few types of tasks in which fractions must be compared and there are a few ways to do this. Usually most students are able to compare fractions with the same denominators because only numerators need to be compared:

$$\frac{3}{5} > \frac{1}{5}.$$

When the denominators are different a student has to find a way to get the result. A student can multiply both sides of the inequality by both denominators:

$$\begin{array}{rcll} \frac{1}{8} & ? & \frac{2}{5} & \mid *5*8 \\ \frac{5*8}{8} & ? & \frac{2*5*8}{5} & \\ 5 & < & 16 & \end{array}$$

A student can also multiply both sides by the least common denominator, but this way needs a skill of factorization:

$$\begin{array}{rcll} \frac{1}{8} & ? & \frac{1}{12} & \mid *24 \\ 3 & > & 2 & \end{array}$$

A student can also move the fraction from the right side of the inequality to the left side leaving zero at the right, then he can make the operation of subtraction and compare the result with zero:

$$\begin{array}{rcl} \frac{4}{5} & ? & \frac{1}{3} \\ \frac{4}{5} - \frac{1}{3} & ? & 0 \\ \frac{7}{15} & > & 0 \end{array}$$

There is one more situation when the task can be solved in two different ways. Having two fractions with the same numerators the student can just compare the fractions without performing any operations but using the fraction imagination - the fraction with a smaller denominator is greater than the other fraction:

$$\frac{3}{5} > \frac{3}{8}.$$

Of course a student can use the reduction of the fractions to the fractions with the same denominator like in general situation. The exercises of comparison which include mixed numbers need an understanding of mixed numbers, and the skill to convert mixed numbers into improper fractions can be included. More about this operation in Section 2.2.7 on page 19.

Comparison in our Student model

We included following skills needed for comparison of fractions to our model:

- *CpD*. Comparison of fractions with the same denominator.
- *CpD-RSD*. Comparison of fractions using reduction to fractions with the same denominator.
- *CpN*. Comparison of fractions with the same numerators.
- *CpD-MNm*. Comparison of mixed numbers using conversion of mixed numbers to improper fractions and reducing them to fractions with the same denominators.
- *CpN-MNm*. Comparison of mixed numbers that have the same whole numbers and the same numerators of proper fractions (e.g. $1\frac{1}{2}, 1\frac{1}{3}$).
- *CpN2*. It was added as an auxiliary node for the CpN exercise.

2.2.3 Addition

Types of addition exercises

This skill can also be divided into a few basic skills. Like in the situation of the comparison addition is easy and understandable when the denominators of the fractions are the same. Then a student needs to know that he must add the numerators and leave the denominator the same:

$$\frac{3}{5} + \frac{1}{5} = \frac{3+1}{5} = \frac{4}{5}.$$

The situation becomes complicated when fractions have different denominators or when they are mixed numbers. In these cases the reduction of the fractions to the fractions with the same denominator is needed:

$$\frac{1}{4} + \frac{1}{3} = \frac{1*3}{4*3} + \frac{1*4}{3*4} = \frac{3+4}{12} = \frac{7}{12}.$$

More about reducing fractions to fractions with the same denominator is presented in Section 2.2.6 on page 19. When fractions include mixed numbers the skill of the understanding of mixed numbers is necessary. More about addition of mixed numbers in Section 2.2.7 on page 19. The result of addition must be a proper fraction (or a mixed number)[2]. A student must reduce the fraction to its lowest terms or to convert it to the compound fraction. A student should have understanding for these operations.

Addition in our Student model

We included following skills needed for addition to our student model:

- *Ad.* Addition of fractions with the same denominators.
- *Ad-RSD.* Addition of fractions using reducing of fractions to fractions with the same denominator.
- *Ad-MNm.* Addition of mixed numbers.
- *Ad-MNm'RLT.* Addition of mixed numbers and reduction of the result fraction to its lowest terms.
- *AdRSD-RLT.* Addition of fractions using reduction to fractions with the same denominator and reducing the result to its lowest terms.

2.2.4 Subtraction

Types of exercises of subtraction

The operation of subtraction is almost of the same difficulty as addition. The easiest exercise is to make the operation of subtraction with the fractions that have the same denominators. A students needs to subtract the numerator of the second fraction from the numerator of the first fraction and to leave the denominator the same:

$$\frac{4}{5} - \frac{1}{5} = \frac{3}{5}.$$

When the denominators of the fractions are different a student needs reducing the fractions to the fractions with the same denominator as in addition (Section 2.2.3). The exercises that include mixed numbers are the most difficult ones in the group of subtraction exercises (Section 2.2.7).

Subtraction in our Student model

We included following skills needed for subtraction to our student model:

- *Sb.* Subtraction of fractions with the same denominator.
- *Sb-RSD.* Subtraction of fractions using reducing of fractions to fractions with the same denominators.

- *SbRSD-RLT*. Subtraction of fractions using reducing of the fractions to fractions with the same denominators and reducing the result fraction to its lowest terms.
- *Sb-MNm*. Subtraction of mixed numbers.
- *Sb-MNmCMI*. Subtraction of mixed numbers by converting the result to a mixed number.
- *Sb-MNm'RLT*. Subtraction of mixed numbers using conversion of the result to its lowest terms.

2.2.5 Multiplication

Types of exercises of multiplication

To multiply fractions a student needs to know the basic rule of multiplication - the product of multiplied numerators must be placed over the product of denominators to get the result[6]:

$$\frac{1}{2} * \frac{3}{5} = \frac{3}{10}.$$

The fraction of the result must be simplified or even converted to a mixed number if it is an improper fraction. Before multiplying fractions it is better to perform canceling of the fractions - to divide the factors of the numerators and the factors of the denominators by the same number:

$$\frac{4}{15} * \frac{5}{12} = \frac{4 * 5}{15 * 12} = \frac{1 * 4 * 5}{3 * 5 * 3 * 4} = \frac{1}{3 * 3} = \frac{1}{9}.$$

Then a student operates with smaller numbers, but the skill of factorization is necessary. If at least one of the fractions in multiplication is compound the exercise becomes more difficult as it needs other skills. More about multiplication with mixed numbers in Section 2.2.7 page 19.

Multiplication in our Student model

We included following skills needed for multiplication to our student model:

- *Mt*. Multiplication of fractions.
- *MtSD*. Multiplication of fractions that have the same denominators.
- *NN*. Understanding that a natural number can be converted to an improper fraction that has a denominator equal to one.
- *MtNN*. Multiplication of the fraction by a natural number.
- *MtMNm*. Multiplication of mixed numbers.
- *Mt-RLT*. Multiplication of fractions using reducing of fractions to their lowest terms.

2.2.6 Reducing fractions with different denominators to the fractions with the same denominator

Where is this skill used?

This is a basic skill as well as fraction imagination. Without this skill a student can not solve most of the exercises concerning fractions - addition, subtraction, comparison of fractions. A student needs to find a common denominator. For example, the easiest way to get a common denominator is to multiply both denominators:

$$\frac{1}{12}+\frac{3}{8}=\frac{1*8}{12*8}+\frac{3*12}{8*12}=\frac{8+36}{8*12}.$$

The find the least common denominator is a more complicated thing. It needs the skill of the number factorization: 12=2*2*3, 8=2*2*2.The common denominator is 2*2*2*3=24.

$$\frac{1}{12}+\frac{3}{8}=\frac{1*2}{12*2}+\frac{3*3}{8*3}=\frac{2+9}{24}.$$

Reducing fractions to fractions with the same denominators in our Student model

We included two skills that correspond to reduction of the fractions to the fractions with the same denominator:

- *RSD*. Reduction of fractions to fractions with the same denominator.
- *UN-RSD*. Understanding that given fractions should be reduced to fractions with the same denominators. More about skills of understanding see in Section 2.3.2.

2.2.7 Mixed numbers

Mixed numbers and basic operations with them

Basic operations with fractions become a bit complicated when the mixed numbers are included. A student must know the rule how to convert the mixed number into an improper fraction (a fraction that has a greater numerator than denominator) - multiply the whole number by the denominator and add the product and the numerator. The result of the sequence of these steps is the numerator of the improper fraction and the denominator is the same as it was in the mixed number:

$$2\frac{3}{5}=\frac{2*5+3}{5}.$$

This action is important in the operations with mixed numbers. But the mixed numbers can be left in addition, subtraction and multiplication, as plus can be imagined between the whole part and the proper fraction of the mixed number. The result of the basic operations (addition, subtraction, multiplication) is usually an improper fraction and it is required to be converted to a mixed number.

- **Comparison of mixed numbers**. When the whole number parts of the mixed numbers are the same then a student needs to compare the proper fractions of the

mixed numbers only as he does in the general situation of comparison (Section 2.2.2):

$$\begin{aligned} 1\frac{1}{2} &> 1\frac{1}{4} \\ \frac{1}{2} &> \frac{1}{4}. \end{aligned}$$

A fraction with a larger whole number part is a larger mixed number and the proper fractions need not be compared:

$$\begin{aligned} 1\frac{1}{2} &< 2\frac{1}{4} \\ 1 &> 2. \end{aligned}$$

The mixed numbers can also be converted to improper fractions and compared using the rules of comparison of proper fractions.

- **Addition and subtraction of mixed numbers**. There are two ways of addition of mixed numbers. The first way is to add the whole numbers, to add proper fractions of the mixed numbers and then to add these to sums (zero can be imagined in the whole part of fractions that have no whole parts):

$$1\frac{3}{5} + 2\frac{2}{3} = (1+2) + (\frac{3}{5} + \frac{2}{3}) = 3 + \frac{9+10}{5*3} = 3\frac{19}{15} = 4\frac{4}{15}.$$

The second way for addition of fractions is to convert the mixed numbers into the proper fractions, add them and then convert the result into a mixed number:

$$1\frac{3}{5} + 2\frac{2}{3} = \frac{8}{5} + \frac{8}{3} = \frac{24+40}{15} = \frac{64}{15} = 4\frac{4}{15}$$

Subtraction of mixed numbers is performed in the same way as addition.

- **Multiplication of mixed numbers**. As it was mentioned before, fractions can be multiplied using conversion of mixed numbers to improper fractions. A student can multiply the improper fractions and then convert the result to a mixed number if it is an improper fraction:

$$1\frac{2}{5} * 2\frac{2}{3} = \frac{7}{5} * \frac{8}{3} = \frac{7*8}{5*3} = \frac{56}{15} = 3\frac{11}{15}.$$

Distributive law also can be used to multiply mixed numbers:

$$(2\frac{1}{5}) * (1\frac{2}{3}) = (2+\frac{1}{5}) * (1+\frac{2}{3}) = 2*1 + 2*\frac{2}{3} + \frac{1}{5}*1 + \frac{1}{5}*\frac{2}{3} =$$
$$= 2 + \frac{4}{3} + \frac{1}{5} + \frac{2}{15} = 2 + \frac{20+3+2}{15} = 2 + \frac{25}{15} = 2 + \frac{5}{3} = 2 + 1\frac{2}{3} = 3\frac{2}{3}.$$

Mixed numbers in our Student model

In our model we included following skills corresponding to the operations with mixed numbers:

- *CMI*. Converting a mixed number to an improper fraction.

- *CIM*. Converting an improper fraction to a mixed number.
- *UN-CMI*. Understanding that a mixed number should be converted to an improper fraction.
- *UN-CIM*. Understanding that an improper fraction should be converted to a mixed number.
- *UMNm*. Understanding of a mixed number as a sum of the whole part and a proper fraction of a mixed number.

2.2.8 Reducing of fractions to fractions with their lowest terms.Expansion

The purpose of reducing the fraction to its lowest terms and the use of expansion

Reducing the fractions to their lowest terms and expansion of the fraction are complementary skills. The result of addition, subtraction or multiplication usually requires simplification - the denominator and the numerator should be factorised and divided by the same number that is called the greatest common factor.Expansion is used when a student adds, subtracts or compares fractions with different denominators. The expansion of the fraction is multiplication of both parts of the fraction by the same natural number.

Reducing of the fraction to its lowest terms in our Student model

We included the skills connected to reducing of the fraction to its lowest terms to our student model:

- *RLT*. Reducing of a fraction to its lowest terms.
- *Ex*. Expansion of a fraction to a fraction with its higher terms.
- *NNFc*. Natural number factorization.
- *Cn*. Canceling of fractions.
- *UN-RLT*. Understanding that a given fraction should be reduced to its lowest terms.

2.3 Structure of our Student model

The structure of our Student model is analysed in this section. The types of the nodes of the model are presented and their descriptions are given.

2.3.1 Types of the nodes

There are six types of nodes in our model (Appendix B):

- Skill nodes;
- Misconception nodes;
- Task nodes;

- Auxiliary nodes (for combining influence of a skill and misconception);
- Logical AND and OR nodes;
- Logical NOT AND nodes.

2.3.2 Skill nodes

This type of node (Figure 2.2) is the main type of node in our Student model, because it corresponds to a skill of a student. Skill nodes can be grouped according to the operations

Figure 2.2: Skill node Sb

to which the skills belong (Section 2.2). Each skill belongs to one of the groups:

- Comparison: CpD, CpD_RSD, CpD_MNm, CpN, CpN2, CpN_MNm
- Addition: Ad, Ad_RSD, Ad_RSD'RLT, Ad_MNm, Ad_MNmRLT
- Subtraction: Sb, Sb_RSD, Sb_MNm, Sb_MNm'RLT, SbMNmCMI
- Multiplication: Mt, MtSD, MtNN, MtMNm, Mt_RLT
- Ability to use a skill: UN_RSD, UN_CMI, UN_CIM, UN_MNm
- Other skills: RSD, CMI, CIM, CN, NNFc, NN

The probability tables for the nodes of this type show the prior beliefs of skills of a student. The probabilities should correspond to to the tested group of students. The experts of this domain can determine the probabilities most precisely. Experience tables are used in the penalized EM algorithm. For example, the probability and experience tables for the node Mt are:

a)

false	0.2
true	0.8

b)

experience	70

Table 2.1: a) Mt probability table and b) Mt experience table

The table means that the probability that students are able to multiply fractions is 0.8. The nodes of the group are connected and they constiture a hierarchy. For example, the node Ad is a parent of the node Ad_RSD, Ad_RSD is a parent for Ad_RSD'RLT and Ad_MNm (see Figure 2.3). This means that if a student has to solve the exercise for Ad_RSD ($\frac{2}{3} + \frac{1}{5}$) then he must have skills

- Ad ($\frac{7}{15} + \frac{3}{15}$)
- RSD (reducing the fractions to the fractions with the same denominator)

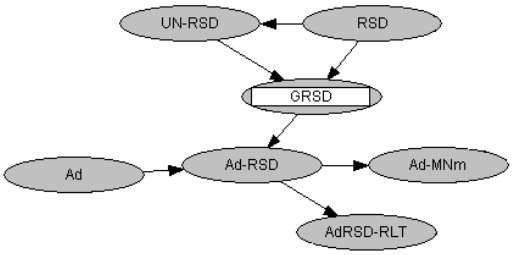


Figure 2.3: Hierarchy of nodes

GRSD	false		true	
Ad	false	true	false	true
false	1	1	1	0.05
true	0	0	0	0.95

Table 2.2: Probability table for the skill Ad_RSD

- UN_RSD (understanding of RSD)
  (the last two skills are combined by the node GRSD - more about this in Section 2.3.6).

The probability table for the skill node Ad_RSD is given in Table 2.2
If a student has the skills GRSD (true) and Ad (true) then the probability that a student has a skill Ad_RSD is equal to 0.95. The probability is not equal to 1 since we want to model that a student will not always be able to solve the exercise even if he has both skills. Ad_RSD requires an additional ability to combine and use the simpler skills. There is one specific feature that makes skill nodes different from the other types of nodes. Skill nodes have experience tables. These tables are used in the penalized EM algorithm for Bayesian network learning (more about EM algorithm in Section 1.4). Table 2.3 is the experience table for node Ad_RSD. The numbers in the experience table mean how certain an expert is

GRSD	false		true	
Ad	false	true	false	true
experience	1000	1000	1000	60

Table 2.3: Experience table for Ad_RSD

about the prior probabilities. The larger number means that experts are more certain. The number values can be understood as a number of observed cases that were used to estimate the probability. Nodes with a prefix UN_ belong to the the group of ability to use a skill nodes. These skills mean that a student is able to use the correspondin basic skill (e.g. UN_RLT means that a student uses his skill RLT). When a student can solve an exercise that requires RSD skill but he does not add the fractions with different denominators - he has not skill UN_RSD. The results of the paper tests approved the necessity of such nodes to be included in the model.

2.3.3 Misconception nodes

The misconception nodes of our model have names that begin with a symbol "-". The nodes

of this type (Figure 2.4) have the same features as skill nodes have. They have experience and prior probabilities. The probabilities tables are updated using the penalized EM algorithm, but misconception nodes do not have any hierarchy. We included these misconceptions to

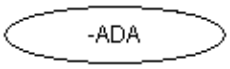


Figure 2.4: Misconception node -ADA

our model (they are described in Section 2.4.3):

- Misconception in addition: -ADA
- Misconception in subtraction: -SDS
- Misconceptions in multiplication: -ANMSD, -MASD, -ADM, -MD
- Misconception in mixed numbers: -MMNm

2.3.4 Task nodes

The task nodes of our model have names that begin with a letter T (see Figure 2.5 for an example of a task node).

Figure 2.5: Task node TMAd

Each task node corresponds to a test item and a task node from an evidence model. Task nodes are separated from the model and our student model is a model without task nodes. These nodes are attached to the model dynamically and the evidence obtained by the student is transmitted to these nodes and after propagation the evidence is transmitted to the evidence model. Table 2.4 is the probability table for the TMAd node.

		Skill Ad		
		false	true	ADA
Task TMAd	false	0.6	0.038	0.038
	true	0.2	0.95	0.12
	ADA	0.2	0.012	0.95

Table 2.4: Probability table for TMAd given Ad

The probabilities in Table 2.4 correspond to a multichoice exercise for the TMAd node with:

- one correct choice
- one choice corresponding to a misconception (in the example it is ADA)
- three false choices.

If a student has skill Ad then

- the probability that he will choose the true answer is equal to 0.95.
- the probability to choose the other answer is 0.05, but there are three possibilities to do this (0.05 is for false answer as the student can make a numerical mistake):
  - the probability to choose a false answer is 0.038 ,
  - the probability that a student has ADA misconception is 0.012 (0.05 was divided by 4).

When a student has a misconception there is a probability equal to 0.95 that he will choose ADA. When a student has not a skill he will probably guess and thus every answercan be chosen with probability 0.2. As there are three false choices then the probability to choose a false answer is 0.6. The probability tables for these nodes are not updated through the network learning.

2.3.5 Auxiliary nodes for combining influence of a skill and misconception

The names of these nodes begin with a letter M (Figure 2.6) in our model (except multiplication nodes Mt, MtSD). These nodes combine skill nodes and misconception nodes so

Figure 2.6: Intermediate node MAd

that each task node has just one parent. This procedure helps to minimize the size of tables for the task nodes which are connected to the model dynamically. The auxiliary nodes have potential tables (Table 2.5). We assume that students can not have both - a skill and a

ADA	false		true	
Ad	false	true	false	true
false	1	0	0	0
true	0	1	0	0.8
ADA	0	0	1	0.2

Table 2.5: Table for the intermediate node MAd

misconception. In the model it is forbidden with a logical NOT AND nodes. Therefore the probabilities for ADA=true and Ad=true are not relevant.

2.3.6 Logical AND nodes

These nodes have names that begin with a letter G in our model (Figure 2.7). These nodes

Figure 2.7: Logical AND node GRSD

combine parent nodes by the AND operation. These nodes help to minimize the number of arrows in the network. For example, GRSD means that a student has RSD and knows how to use the skill. GRSD node has a conditional probability table given in Table 2.6. Logical AND nodes have no experience tables.

RSD	false		true	
UN_RSD	false	true	false	true
false	0	0	0	1
true	1	1	1	0

Table 2.6: Table for GRSD node

2.3.7 Logical NOT AND nodes

These nodes have names that begin with "!" in our student model (Figure 2.8). As it

Figure 2.8: Logical NOT AND node !NANDCMI

was mentioned above a student can not have a skill and a misconception for the skill at the same time. We have !NAND nodes to avoid such situations and use NOT(AND) operation (Table 2.7). A student can not have -MMNm and CMI at the same time when NAND has

-MMNm	false		true	
CMI	false	true	false	true
false	0	0	0	1
true	1	1	1	0

Table 2.7: Table for NAND_CMI

the evidence true. The evidence for NAND nodes is inserted during the initialization of the tests - there is a special node _!_ for this purpose and it is initialised immediately after loading the network. These nodes have no experience tables.

2.4 Paper tests

This section deals with the paper tests we gave to the students of a secondary school. The purpose of these tests is explained. The process of the preparation and the contents of the tests are described. The analysis of the results is given.

2.4.1 Preparation of the tests

Having decided to choose the domain of the fractions we needed an advice of someone who had experience in working with children who learn fractions. We had an appointment with a teacher of mathematics working in the secondary school in Brønderslev. She gave us the information according to which we worked further. We got the understanding about the level of students' knowledge in fractions. During the meeting with the teacher it was agreed

Skill	3a,b	4a,b	5a	5b,c	6a,b	7a,b	8a	8b	8c	9a	9b	9c	10a	10b	10c
RLT	+							+		(2)[1]			(2)		
RSD		+													
TCpD_3			+												
TCpD_RSD_3				+											
CMI					+				+			+			+
CIM						+			+			+			+
MtSD							+								
Mt								+	+						
Mt_RLT								+							
UN_RLT								+		(2)			(2)		
MtMNm									+						
UN_CMI									+			+			+
UN_CIM									+			+			+
Ad										+	+	+			
Ad_RSD											+				
UN_RSD											+			+	
Ad_MNmRLT												+			
Sb													+	+	+
Sb_RSD														+	
Sb_MNm															+

Table 2.8: Skills tested in the exercises

that we would create the test and the teacher would give it to her students as we needed the information about skills of students for our model. We created two tests and each of them had 10 groups of the exercises (see the tests in AppendixA). We chose the exercises according to the information we got from the teacher and we expected the students to have some misconceptions.

We wanted to know how students imagine fractions. That is why the first exercise asks students to describe how they understand a given fraction. The second exercise asks to choose one of the two pictures or to draw the other one to explain what the fraction is in the student's mind. We expected that students think of fractions as of a part of a pie or the part of x-axis.

The third group of exercises asks to reduce the fractions to their lowest terms. This skill is usually required after the operation of addition or subtraction. The forth exercise asks to find the least common denominator for the fractions as this skill is important in addition and subtraction when fractions do not have the same denominators.

The fifth group of exercises asks to compare fractions - to decide which one is greater than the another one.

The sixth and the seventh groups are related to the compound fractions. The sixth group asks to transform the compound fractions into improper fractions and the seventh asks to do it vice versa.

The last three groups of exercises are related to basic operations - multiplication, addition and subtraction. There are three exercises for each of them. The first one includes fractions with the same denominator, the second one includes fractions with different denominators and there is a compound fraction included in the third one. Table 2.8 on page 27 shows which skills were tested in each exercise. The tests were given to the students of 15 years old in the secondary school in Brønderslev. 149 students did our tests.

[1] These skills were necessary only in Test 2

2.4.2 Analysis of the paper tests

We have made the analysis of results students got making our tests (see in Appendix C).

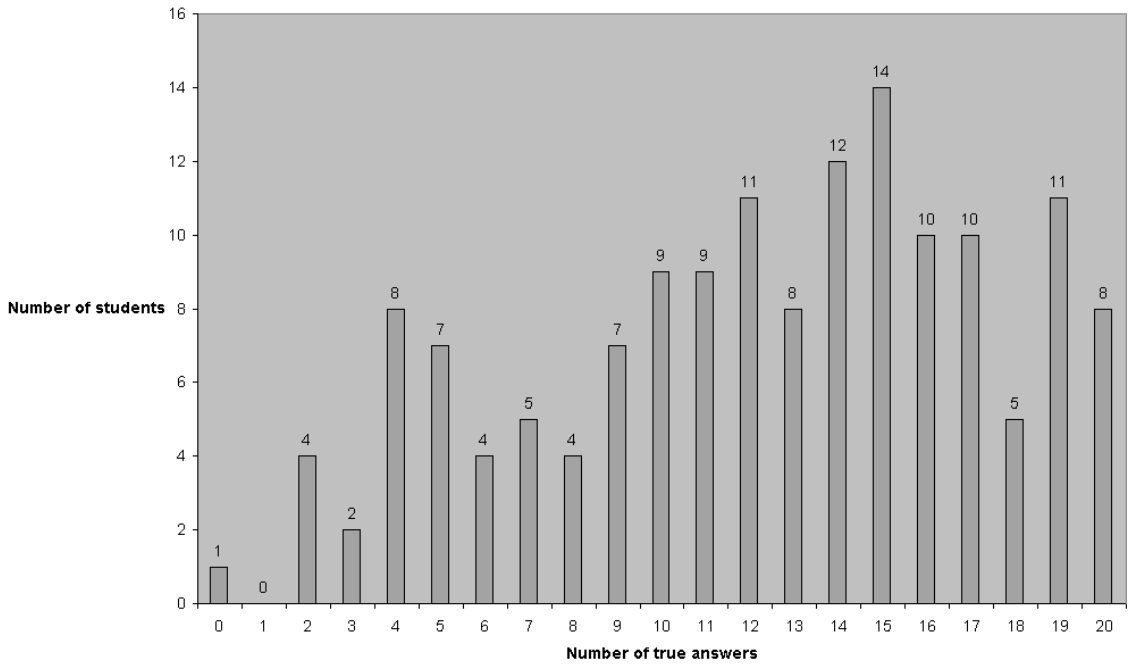


Figure 2.9: Distribution of the students according to the number of true answers

Complexity of our tests

Figure 2.9 shows how many students were able to solve different number of exercises. According to the statistics we can say that we prepared the tests for the students of the medium level of knowledge in fractions. We had 20 exercises.

- 19 students solved more than eighteen exercises;
- 7 students solved less than four exercises.

This means that our tests were not difficult, but also they were not easy and not all the students were able to do all the exercises:

- 98 students (65.77%) made more than ten exercises (i.e. 50%).
- 44 (29.53%) of those 98 solved more than fifteen exercises (i.e. 75%).

Groups of exercises

The first two exercises were used to clear up how students imagine fractions. Most students imagine fractions as a part of a pie. We defined the prior probabilities for the Img node in our network according to this information. We learned what exercises of our tests were difficult for students (Table 2.10). Most students know how:

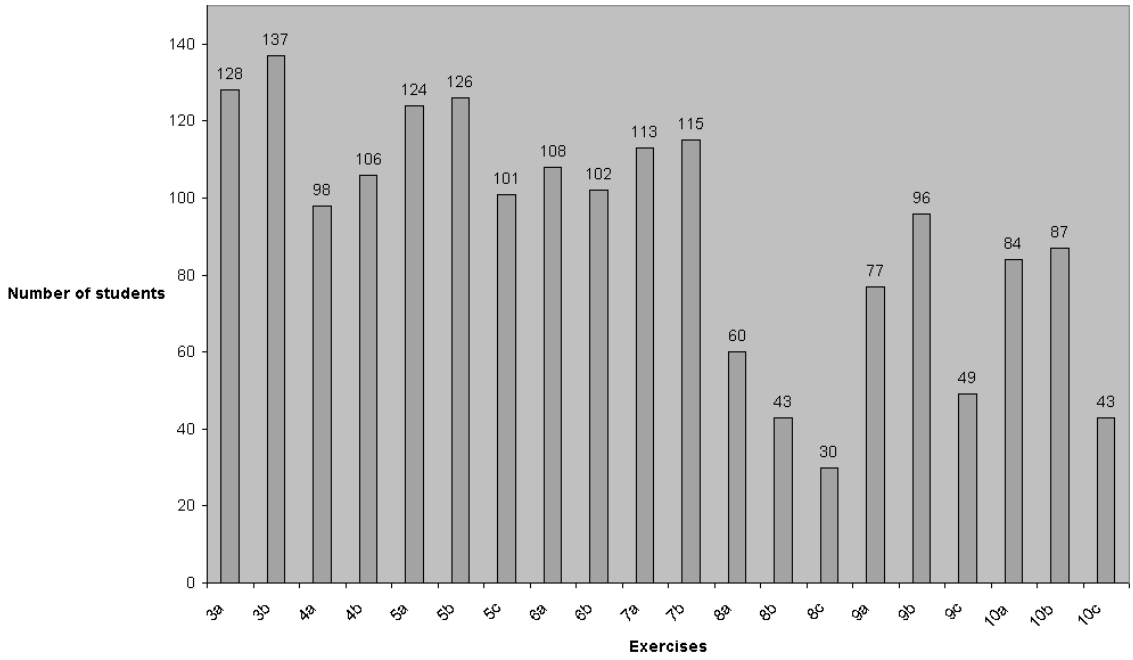


Figure 2.10: Statistics for each exercise

- to reduce a fraction to its lowest terms (exercises 3a,b);
- to reduce the fractions to the fractions with the same denominators (exercises 4a,b);
- to compare the fractions (exercises 5a, b, c);
- to convert a mixed number to an improper fraction (exercises 6a, b) or convert an improper fraction to a mixed number (exercises 7a, b).

But much less of them are able to combine these skills in more complex exercises (groups 8, 9, 10). The number of students who solved the exercises 8-10 is lower than the number of students who solved exercises 3-7. The groups of exercises 8-10 were organized to give different situations for the same operation. We see from the table that the most complex exercises were those which included mixed numbers (8c, 9c, 10c). The students are able

- to convert a mixed number to an improper fraction (ex.6: a - 108, b - 102),
- to convert an improper fraction to a mixed number (ex.7: a - 113, b - 115).

But they do not understand how operate with mixed numbers:

- multiply (ex. 8c- 30),
- add (ex. 9c - 49),
- subtract (ex. 10c - 43).

Exercise 9a required addition of two fractions with the same denominators and 10a required subtraction of two fractions with the same denominators. It seems they were more difficult for the students than the exercises that included the fractions with different denominators (9b, 10b):

- 9a - 77 and 9b - 96;
- 10a - 84 and 10b - 87.

The students know that many rules exist for the situation when the denominators are the same but they do not remember exactly what to do. The most difficult operation for students is multiplication. We found five misconceptions that are connected with this operation (see Section 2.4.3).

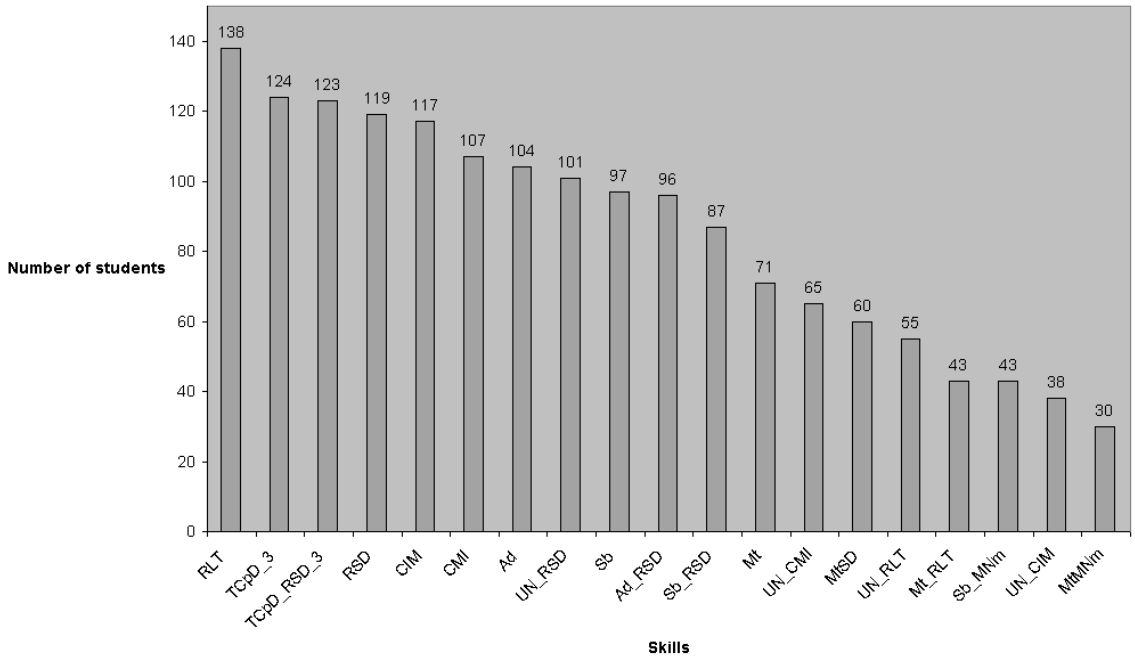


Figure 2.11: Skills

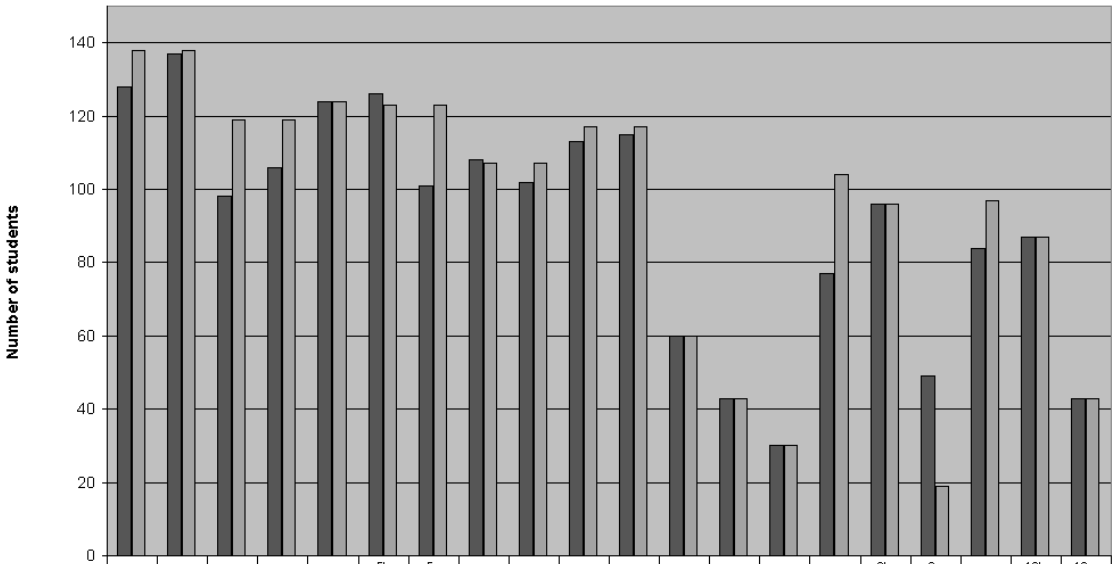
Skills

Figure 2.11 shows how many students have each skill.

- 138 students have the skill of reducing the fraction to its lowest terms (RLT)
  - but only 55 of them are able to use this skill in complex exercises (UN_RLT)
  - and only 43 students can multiply a fraction(Mt_RLT) or add mixed numbers (AdMNmRLT) and reduce the result fraction to its lowest terms.
- 107 students can convert a mixed number to an improper fraction(CMI) but only 65 are able to use it(UN_CMI). The same thing happens with conversion of an improper fraction to a mixed number(CIM) - only 38 students understand how to use this skill (UN_CIM).
- The number of students that have the skill of reducing fractions to fractions with the same denominator(RSD) and the number of students that have an understanding of using (UN_RSD) it do not differ much:

- 119 students have RSD,
- 101 students have UN_RSD.

The lowest numbers are for the skills that are connected with mixed numbers - multiplication(MtMNm), addition(Ad_MNmRLT) and subtraction (Sb_MNm).



	3a RLT	3b RLT	4a RSD	4b RSD	5a TCpD_3	5b TCpD_RSD_3	5c TCpD_RSD_3	6a CMI	6b CMI	7a CIM	7b CIM	8a MtSD	8b Mt_RLT	8c MtMNm	9a Ad	9b Ad_RSD	9c Ad_MNmRLT	10a Sb	10b Sb_RSD	10c Sb_MNm
Questions	128	137	98	106	124	126	101	108	102	113	115	60	43	30	77	96	49	84	87	43
Skills	138	138	119	119	124	123	123	107	107	117	117	60	43	30	104	96	19	97	87	43

Figure 2.12: Exercises compared with the main skill

In Figure 2.12 a pair of columns for each exercise is presented. The first column represents the number of students that solved an exercise correctly. The second column represents the number of students that showed they have the skill necessary to solve the exercise. Some skills were tested using a few exercises and we made the conclusions according to their results (Table 2.8). For example, RLT was tested in exercises 3a,b, 8b and 9a,10a in the second test. For each exercise there is a main skill - RLT is main for 3a and 3b, and Mt_RLT is main for 8b. As it can be seen from the diagram there are a few big differences between the number of students that solved an exercise and the amount of students that have the main skill for the corresponding exercise. For example, 9a has the main skill Ad (addition):

- 77 students solved the exercise correctly,
- but 104 students have the skill of addition. Some students solved the other addition exercises correctly and did a numerical mistake in this one (or did not finish the exercise as we required) - however they proved that they do have the skill of addition.

The columns for 9c (Ad_MNmRLT) show an opposite situation. The amount of students that have the skill is lower than the amount of students that solved the exercise 9c. This situation appeared as some students solved the exercise correctly, but we could not decide if they have a skill or not. They got the correct result but according to other related exercises they do not seem to have this skill - the students did not solve easier exercises of addition, so perhaps they have a misconception. This unclear situation was left not resolved.

Nr	Name	Example	Pattern	Cases	Percentage
1	ADA	$\frac{1}{3} + \frac{1}{3} = \frac{2}{6}$	$\frac{a}{x} + \frac{b}{x} = \frac{a+b}{x+x}$	22	14.8
2	SDS	$\frac{2}{3} - \frac{1}{3} = \frac{1}{0}$	$\frac{a}{x} - \frac{b}{x} = \frac{a-b}{x-x}$	14	9.4
3	MD	$\frac{1}{4}$ X $\frac{3}{4} = \frac{1*4}{4*3} = \frac{4}{12}$	$\frac{a}{x} * \frac{b}{y} = \frac{a*y}{x*b}$	23	15.4
4	MASD	$\frac{1}{4} * \frac{3}{4} = \frac{1*3}{4} = \frac{3}{4}$	$\frac{a}{x} * \frac{b}{x} = \frac{a*b}{x}$	21	14.1
5	MMNm	$1\frac{1}{2} = \frac{1*1}{2} = \frac{1}{2}$; $1\frac{1}{2} * \frac{1}{4} = \frac{1}{8}$	$A\frac{c}{d} = \frac{A*c}{d}$	6	4.0
6	ANMSD	$\frac{2}{3} * \frac{1}{3} = \frac{2+1}{3*3} = \frac{3}{9}$	$\frac{a}{x} * \frac{b}{x} = \frac{a+b}{x*x}$	12	8.1
7	ADM	$1\frac{1}{2} * \frac{1}{4} = \frac{3*1}{2+4} = \frac{3}{6}$	$\frac{a}{x} * \frac{b}{y} = \frac{a*b}{x+y}$	12	8.1
8	ISMNm	$1\frac{3}{6} - \frac{4}{6} = 1\frac{1}{6}$		4	2.7
9	AWNmN	$2\frac{2}{3} = \frac{2+2}{3}$	$A\frac{c}{d} = \frac{A+c}{d}$	1	0.7
10	MAPM	$\frac{1}{4} * \frac{4}{5} = \frac{80}{20}$	$\frac{a}{b} * \frac{c}{d} = \frac{(a*d)*(c*b)}{b*d}$	6	4.0

Table 2.9: Misconceptions

2.4.3 Misconceptions

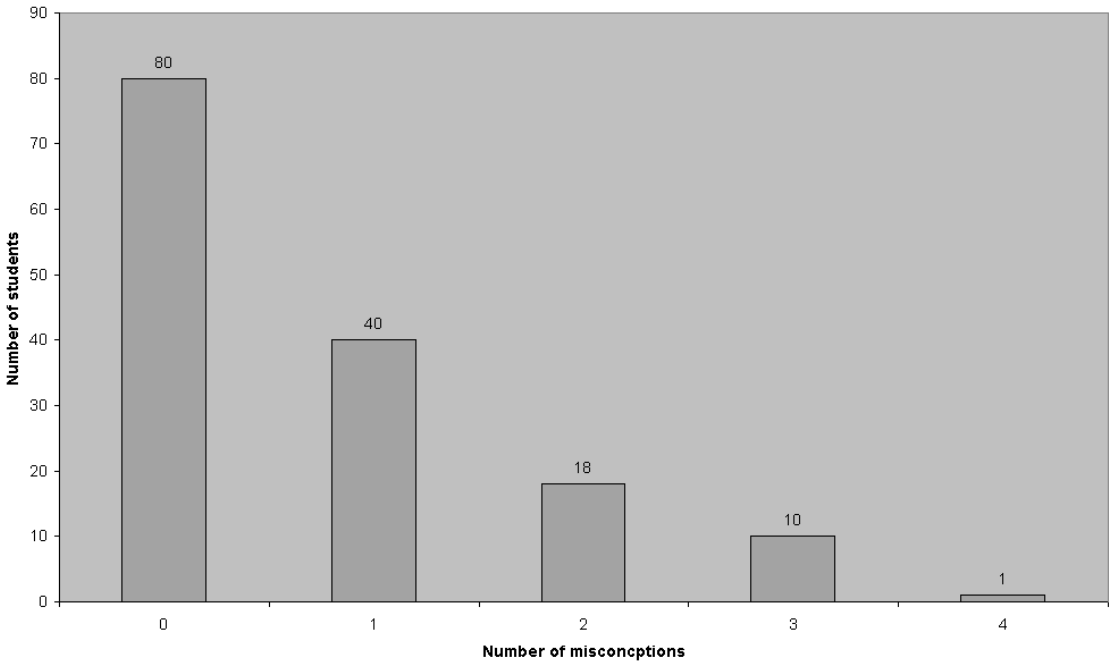


Figure 2.13: Distribution of students according to the number of their misconceptions

During the analysis of the results of students we found nine misconceptions (Table 2.9). They can be divided into four groups.

- Two misconceptions (ADA, ISMNm) were found in the exercises which included the operation of addition.
- One misconception (SDS) was found in the exercises of subtraction.
- Two misconceptions appeared when students solved the exercises which included mixed numbers (MMNm, AWNmN).

- The largest group of misconceptions is connected with multiplication - five misconceptions (MD, MASD, ANMSD,ADM, MAPM).

The patterns, examples, statistics of misconceptions are given in Table 2.9.
Figure 2.13 shows the distribution of students according to the number of misconceptions they have.

- 80 students (of 149) have no misconceptions (53.69%);
- 40 students have one misconception (26.85%).

We see that 80.54% of students have less than two misconceptions. Only one student has four misconceptions out of nine and 10 students have three misconceptions - just 7.38% of students have more than two misconceptions. The left 12.08% of students have two misconceptions. We can make the conclusion that even if we found nine misconceptions in the papers - not many students have them. We have about 54% of students with no misconception, but Figure 2.9 on page 28 shows that just 8 students solved all the exercises. This means that either students do not have misconception, either they did not try to solve the exercise or they did a numerical mistake.

ADA

Addition of **D**enominators in **A**ddition. This misconception appeared in the exercises of addition. Students added denominators as well as numerators. ADA is one of misconceptions that appeared frequently - 22 students (14.8%) have it (see Table 2.9).

SDS

Subtraction of **D**enominators in **S**ubtraction. The pattern of this misconception is almost the same as in ADA, but misconception is connected to subtraction and students subtracted denominators as well as numerators.

MD

Multiplication using **D**ivision rules. This misconception appeared when students mixed the rules of multiplication with division rules. The students multiplied the numerator of the first fraction by the denominator of the second fraction and the denominator of the first fraction by the numerator of the second. It means that the second fraction was turned over as in division. MD is the most frequent misconception - 23 students have it (15.4%).

MASD

Multiplication like **A**ddition with the **S**ame **D**enominator. Students mixed addition with multiplication. When fractions had the same denominators in the exercises of multiplication students having this misconception left the same denominator for the result instead of raising it to the second power. They multiplied only the numerators.

MAPM

Multiplication of **A**ll **P**arts of fractions in **M**ultiplication. The misconception is connected to multiplication. Before multiplying fractions the students reduced the fractions to the fractions with the same denominator, they left the common denominator for the result but multiplied numerators.

MMNm

Multiplication in **M**ixed **Nu**m**b**ers. The misconception was found in the exercises which included mixed numbers. Students multiplied the whole part of the mixed number by the numerator.

ANMSD

Addition of **N**umerators in **M**ultiplication of fractions with the **S**ame **D**enominator. This misconception was found in the exercises of multiplication. Students multiplied denominators, but they added numerators instead of multiplying them.

ADM

Addition of **D**enominators in **M**ultiplication. This misconception is complementary to ANMSD. Students multiplied numerators but added denominators.

ISMNm

Improper **S**ubtraction of **M**ixed **Nu**m**b**er. The misconception is connected to the understanding of mixed numbers. Students left the whole part of the first fraction, then subtracted proper fractions, got the negative fraction but they wrote it as if it was positive.

AWNmN

Addition of the **W**hole **Nu**m**b**er and the **N**umerator in mixed numbers. This misconception was found just once. The student converted the mixed number using his own rule - he added the whole number with the numerator.

2.4.4 Brief summary of the results of the paper tests

Having analysed the paper tests we added the misconceptions to our student model. We validated the structure of our network. We got the prior probabilities for our network. Before testing students we doubted if we should add skills of understanding what skills use and how in an operation(e.g. in the operations like reduction of a fraction to its lowest terms or conversion of a mixed number to improper fraction). Since many students who had the necessary skills were not able to use them properly we included the "understanding" skills in our network. We found nine misconceptions but to have more detailed analysis special test would be needed.

2.5 Learning network of our Student model

This section deals with the Bayesian network learning. The learning procedure is described and the results of the learning verification are presented.

2.5.1 Process of the learning

We prepared the paper tests and gave them to the students of a secondary school in Brønderslev. We tested the students of 15 years old and we used this information for the learning network. As it was mentioned earlier (Section 2.3.2) the probability tables for skill nodes can be filled by the domain expert. An expert working with students can help us to find the prior probabilities. The model should be adapted to the group of students the tests are created for. Another way to get the probabilities is to prepare paper tests for the students and find out the necessary probabilities from the results. On the basis of the testing results it is also possible to determine the misconceptions the students have and to find exercises most difficult for the students. According to the results the network learning is performed using EM algorithm. For the research of the group of the students we have consulted a secondary school teacher and prepared paper tests (more about paper tests in Section 2.4.1). We summarized the results as a list of data vectors (Appendix C). We used the penalized EM algorithm (described in Section 1.4) implemented in Hugin Decision Engine (HDE) for the network learning (more about HDE in Section 3.3). To learn each node EM algorithm uses the experience tables of nodes (see Section 2.3.2). We do not need to update probabilities for logical type nodes. So these nodes do not have experience tables. For auxiliary nodes we should update only one column which corresponds to the situation when all the parents of a node have values true (Table 2.2). Other columns should stay unchanged because we are sure about values in them. Experience values in these columns are equal to 1000.

2.5.2 Verification of the learning network

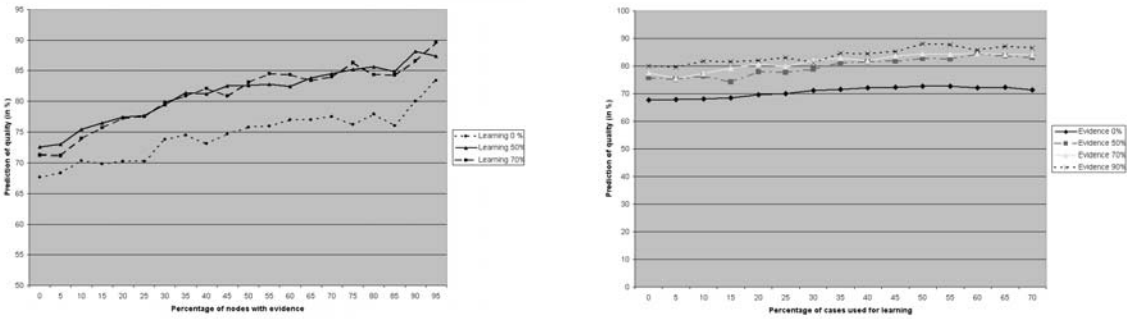
To test the quality of the learned Bayesian network we used the results from the paper tests. The results were divided into two parts: 70% of the test results (chosen randomly) were used learn the Bayesian network and 30% were left to imitate students. We wanted to see how well our Student model can predict the answers of each student from the group of students we tested. Each tested case is a set of skills and misconceptions of a student.

In the sequel we will use the term skills for both skills and misconceptions. The t observed skills are chosen randomly from all $n = 20$ skills and the evidence is inserted into the student model. The student model knows t skills of a student and it guesses the other $n - t$ skills. The bigger number of the correct guesses shows the better correspondence of the model to a student. The guess is correct if the probability of the answer in the data is higher than 0.5 in the model. In each paper test we tested 20 skills of a student. For example, we choose randomly $t = 15$ skills and enter the evidence for these skills into the model. The left $20 - t = 5$ skills are used to check the correspondence of our model to the paper test. We read the belief for every skill out of these five skills from our model. If the belief, for example, is $P(true) = 0.6, P(false) = 0.4$ and from the paper test we read that a student has the skill then we consider the model to guess correctly. If a skill$= false$ in the model then we consider the model to predict wrong. The tests were done for various values of t

corresponding to $e = \frac{t}{n}$, $e \in \{0\%, 5\%, 10\%, ..., 95\%\}$. The average of 10 runs of the test with the same parameters reflect the dependency between the amount of evidence included in the model and the number of correct predictions of the remaining $n - t$ skills. The tests were done on network:

- before learning tree,
- after learning of 50% of cases,
- after learning of 70% of cases.

The results are displayed in Figure 2.14(a).



(a) Verification according to entered evidence

(b) Verification according to learning

Figure 2.14: Verification of the network

The graph in Figure 2.14(a) shows that the model has 50% prediction accuracy even before learning and without any evidence. According to the first curve learning 0% our model predicts answers of the tested group of students quite well even without learning. The curve that represents the model after learning 70% of the cases (we left 30% of randomly selected items from the item bank to imitate students) shows better results. The student model corresponds to the group of students better after learning than before it. After learning procedure the quality of prediction increases approximately 10 - 15%. The difference is quite small and some reasons can be found:

- We observed only 149 student cases;
- After test analysis we got more knowledge about the student group and we manually corrected some prior probabilities according to the test results.

The learned model was tuned with respect to the group of students that were tested but it is possible that it will not model well the other group of students. To make the student model general the data from various groups of students should be used.

Chapter 3

CAT implementation

3.1 Decription of the program for the Computer Adaptive Test of Fractions

In this chapter, we discuss the implementation of the program. We give the class diagram of the CAT program and a description of the structure of the item bank. Basic guidelines of the program usage and a short description of the Hugin Decision Engine(HDE) used in the program are presented.

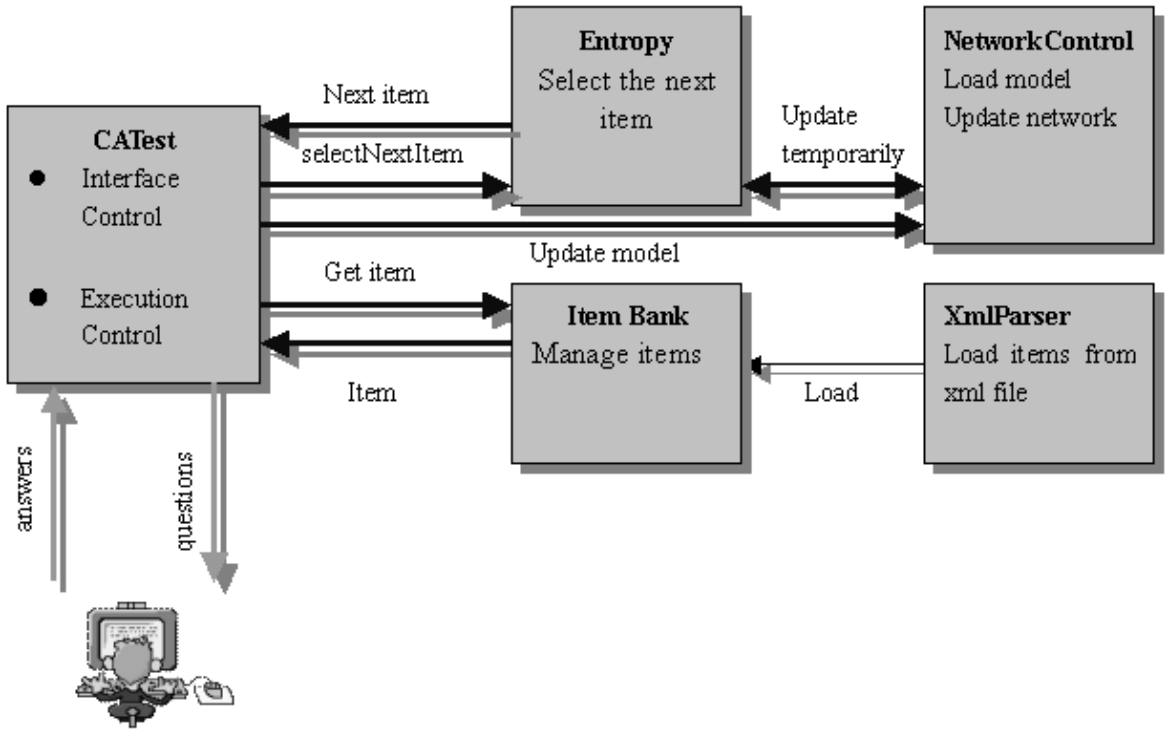


Figure 3.1: System architecture

The program consists of five main parts. Each part is responsible for a particular task. The CATest is the main part, it takes control over all other parts. This part also handles the

operations concerning the user interface. The XmlParser loads the ItemBank from the xml file. The ItemBank manages Items. The NetworkControl handles the operations concerning the Bayesian Network. The part of the Entropy controls the order of appearance of items in the test.

3.1.1 Class diagrams and description of main classes

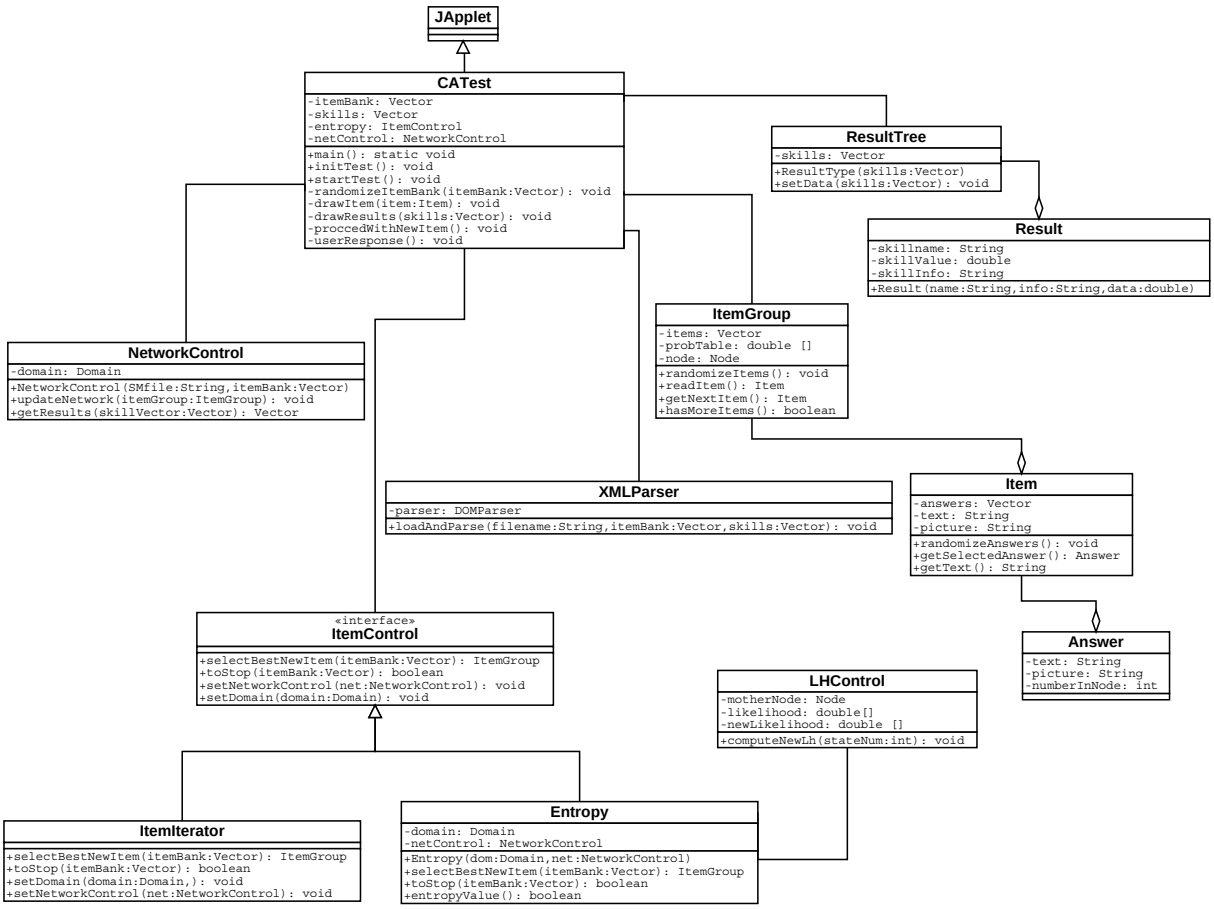


Figure 3.2: Class diagram

Only the most important methods and attributes are included in this class diagram.

class CATest

The class CATest is the main class, which controls the execution of the whole program. It provides the user interface and responses to the user's selection of answers.
The main methods are:

- `initTest()`

  initializes the test, loads the student model and creates the item bank

- `randomizeItemBank()`

  mixes ordering of Items in each ItemGroup

- `proccedWithNewItem()`

  finds the most informative item using the Entropy object, and gives this item to a student.

- `userResponse()`

  handles the response of a user and passes the evidence to the NetworkControl object

interface ItemControl

The interface ItemControl is used as the general interface for the item selection procedure. The main methods are:

- `ItemGroup selectBestNewItem(Vector itemBank)`

  has to return the next item in the test

- `boolean  toStop(Vector itemBank)`

  has to return true if the test should be finished

class Entropy

The class Entropy handles mathematical calculations. This class finds the item to proceed next. The main methods are:

- `ItemGroup selectBestNewItem(Vector itemBank)`

  is used to find the item in the itemBank. This method uses the method described in section 1.6

- `boolean  toStop(Vector itemBank)`

  this method returns true if the test should be finished.

- `double  entropyValue()`

  this method calculates the entropy of the probability distribution of the Student Model

class ItemIterator

The class ItemIterator is used to go through all items in the item bank, without any clever selection algorithm. This class is used as a tool to find errors in the itemBank. The main methods are:

- `ItemGroup selectBestNewItem(Vector itemBank)`

  returns the item from itemGroup item by item.

- `boolean  toStop(Vector itemBank)`

  this method returns true if no more items have left in the item bank

Class NetworkControl

This class is mainly used to load the student model, build the evidence model, and update the network. When the program starts, Class NetworkControl parses the Bayesian Network file which has the extension "hkb", and loads the network into a domain, that is the way how student model is built. Because the item nodes in the evidence model won't be taken into consideration when we calculate the entropy value, the NetworkControl records the cliques and intersections in the student model and store them into two Vectors. The number of them are fixed during the test. Then, the NetworkControl builds the evidence models, it reads the infomation for each item node from Class Item. The information includes parent nodes' names of each item, the number of its states, the conditional probability table, etc. All the Hugin nodes will be created and attached to the student model. Thus the initialization part ends. During the test, Class NetworkControl will modify and update the network by the requests from the other parts of the program.

- `NetworkControl(String SMfile, Vector itemBank)`

  is the constructor for NetworkControl. It creates the domain object, loads the student model from ".hkb" file and builds the evidence models using the data of the itemBank.

- `void updateNetwork(Vector itemBank)`

  updates the network by performing the soft evidential update instead of the hard one (please read Section 1.5 for the definitions of hard and soft evidential update). However hard evidential update is performed during the process of the soft evidential update, but it is retracted when the updated values were read. The goal is to keep a skill node updated by outcomes of all answered children items.

- `Vector getResults(Vector skills)`

  The task for this method is to read the current beliefs from the student model. This method is called after the test is finished.

class ItemGroup

The class ItemGroup is a storage class for items of the same type. Items of the same type have the same parent node in the student model and the same probability tables. The main methods are:

- `Item readItem()`

  reads the current available item from the item group.

- `Item getNextItem()`

  reads the next item from the item group and this item becomes the current item in this item group.

- `void randomizeItems()`

  mixes the ordering of items in the ItemGroup. The initial ordering is the same as in the xml file.

class Item

The class Item stores information related to one test item. The main methods are:

- `String  getText()`

  returns the question of an item.

- `Answer getSelectedAnswer()`

  after a student selects one possible answer for an item, the selected answer is provided by this method

- `void randomizeAnswers()`

  mixes the ordering of answers in an Item. The initial ordering is the same as in the xml file.

class XMLParser

The class XMLParser loads the itemBank and descriptions of skills from the xml file. The main methods are:

- `void loadAndParse(String fileName, Vector itemBank, Vector skills)`

  this method reads the file with a name fileName and fills the data from this file into two Vectors.

3.1.2 Xml file of Item bank

The item bank is stored in an xml file. The structure of the xml file is given below.

```
<itembank>
 <skillsdescriptions>
  <skill type="skill">
    <nodename>MMt</nodename>
    <description>This skill represents multiplication</description>
  </skill>
   ...
 </skillsdescriptions>

 <itemgroup>
  <parents>
   <parent>First.parent</parent>
   ...
  </parents>
  <table>
   <probab id="yes"  number="2" conf1="0.1" conf2="0.4"/>
   <probab id="no"   number="2" conf1="0.2" conf2="0.5"/>
   <probab id="misc" number="2" conf1="0.3" conf2="0.6"/>
   ...
  </table>
```

```
  <item>
    <task>
     {<picture {width="20"|[0]} {height="30"|[0]}>
       resources/images/img.gif
      </picture>|[""]}
     <text {layout="left"|["right"]}>
        layouts can be left,right,top, bottom
     </text>
    </task>
    <answers>
     <answer>
      {<picture {width="10"|[0]} {height="5"|[0]}>
           resources/images/img2.gif
       </picture>|[""]}
      <text {layout="left"|["right"]}>answer a</text>
      <probab id="yes"/>
     </answer>
     ...
    </answers>
  </item>
  ...
 </itemgroup>
 ...
</itembank>
```

In this example a specific notation is used to explain possible alternations in the xml file. The meanings of symbols are:

- xml element between

  ```
  { xmlItem |[""]}
  ```

  like

  ```
   {<picture> </picture>|["fraction.gif"]}
  ```

  means, that a picture of the element is not required and if no picture of the element is present the value for the attribute picture in the object Item will be set to "fraction.gif".

- three points like in this example

  ```
    </skill>
     ...
   </skillsdescriptions>
  ```

  mean that many elements of type <*skill*>can be placed in the element <*skillsdescription*>.

In the beginning of the xml file all skills with their descriptions are placed. These descriptions are used in the window with test results (Figure 3.3(c)). After all <*itemgroup*> elements are listed. Each element <*itemgroup*> has its node name from the student model, to which the task item is connected. The element <*table*> has a conditional probability table for the task node. Each row of the table has a specific name in atribute *id*. Each row represents the row of a state in the conditional probability table of the corresponding node in the Bayesian network. The attribute *number* has the length of the row. Further in <*itemgroup*> all items are listed. Each element <*item*> has a text of the corresponding question in the element <*text*> and a name of an attached picture - in element <*picture*>. Further in element <*item*> all answers are listed. Element <*answer*> should have an association to a row of the table, this association is represented by the row id.

3.1.3 Some usage guidelines

Our program was written in Java. To run it from the command line write

```
run.bat
```

(**Hugin API v5.2** or higher must be installed).This script file, sets environment variables and runs Java VM.

The program can be started in two modes: debug mode and normal mode. In the debug mode each answer near its radio button will have a row id from the xml file(Figure 3.3(b)). To start the debug mode keywords should be added:

- */d* - with this key the program will start in the debug mode (figure 3.3(b))
- */di* - with this key the program will debug all items in the item bank
- */?* - displays possible keys

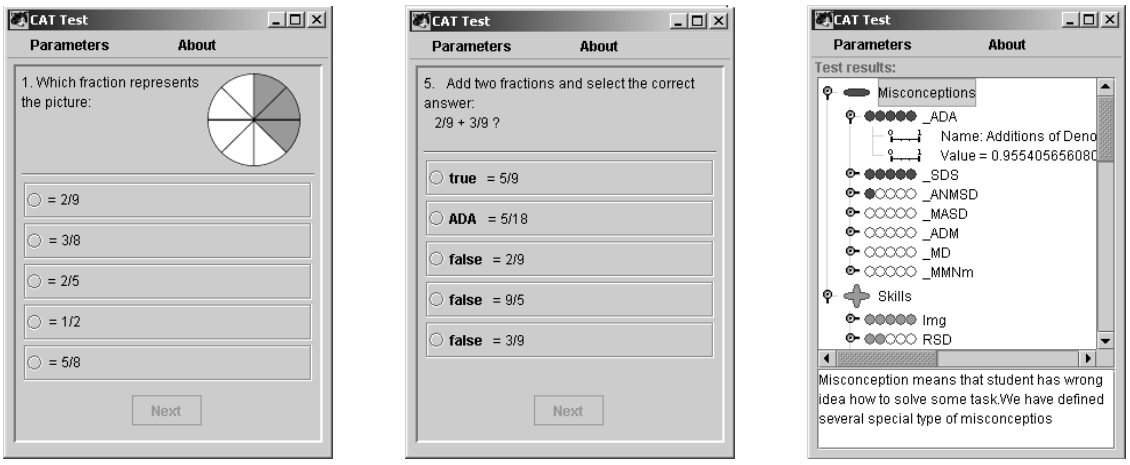
3.2 Performance of item selection procedure

To verify performance of the item selection procedure, we have performed several tests. In these tests we observe changes of an entropy value according to selected (and answered) items by item selection (see Section 1.6) and random selection procedures.

Figure 3.4 shows results for tests performed using the item selection procedure described in Section 1.6 and selecting correct, incorrect and random answers for each given item.

Figure 3.5 shows results for test performed using random item selection and selecting correct answers for each given item.

As we can see entropy value using selection method described in Section1.6 decreases much faster with comparison to random ordering (random ordering have some similarities with paper test). This shows that using item selection procedure, we can faster determine student abilities.



(a) Test page

(b) Test page in the debug mode

(c) Result page

Figure 3.3: Screenshots

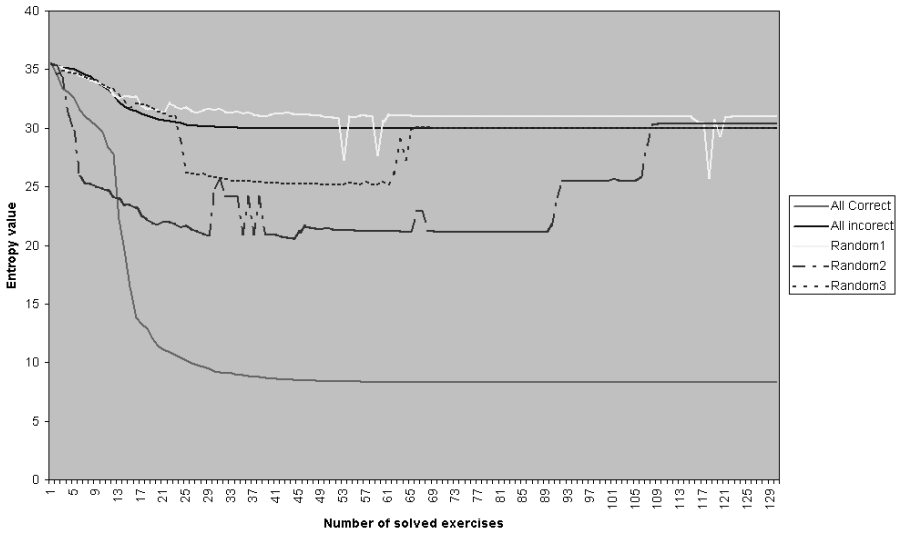


Figure 3.4: Entropy values selecting with method `selectBestNewItem`

3.3 Introduction to the Hugin Decision Engine (HDE)

The CAT is an applet running on the Hugin Decision Engine(HDE). The HDE is delivered with the application program interfaces (API's) for four major programming languages C, C++, Java and an ActiveX-server for e.g. Visual Basic.

The HUGIN API (application program interface) is a library that allows a programmer to use Bayesian belief networks (with both discrete and continuous variables) and influence diagrams in his own application. It is used as most other libraries: the application program invokes HUGIN API functions whenever it wants to have operations of the belief networks performed. [4]

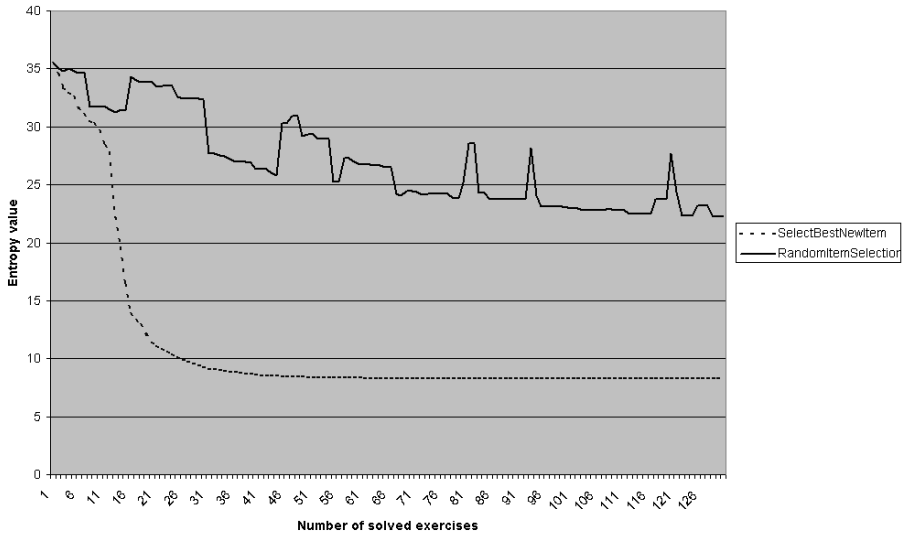


Figure 3.5: Entropy values randomly selecting item

3.3.1 Used features

1. **Construction of models**

   Major functions used for model construction are described in this section.

   - saving/loading of domains. The program first creates a Domain object by loading a file with an extension "hkb" which is a representation of belief networks. While the program exits, it closes and saves the same file.
   - accessing the probability of a state of a node (read belief).
   - define dependence. Function Node.addParent() makes it possible to define dependence between two nodes.
   - editing the table data (probabilities). After attaching a node to one or more nodes, a table of the child node is created.
   - accessing/changing the name, labels, the number of states, names, labels of a node. The manipulating of the attributes of nodes is important for creating a node or gathering the information. For example, users can search a particular node by Domain.getNodeByName(String).

2. **Usage**

   Functions described in this section are mostly used with already constructed models.

   - insertion of both hard (instantiation) and soft (likelihood) evidence. Hard evidence means a node is instantiated by selecting a certain state while the soft evidence represents that a more general item of evidence called "likelihood" can be entered into discrete nodes. Users may update the network by entering either soft evidence or hard evidence.
   - compilation of a domain. Before a belief network can be used for the inference, it must be compiled. The work such as selecting a state of a node, propagation could only be done after the network is compiled.

- retraction of evidence. By this feature, users can retract an already entered observations, return the node to its previous statues. We use this in calculating the entropy of the network.
- propagation of evidence in the junction tree of cliques. The probability of the propagated evidence is available as a result of propagation (normalization constant). After the evidence has been entered into a domain, we want to get the revised beliefs for some or all nodes of the domain. This is done by incorporating the specified evidence into the junction tree potentials and performing a propagation operation on the junction tree(s).
- computation of the joint probability distributions of a set of discrete variables. Using Domain.getMarginals(NodeList), users can easily get the probability table of a list of nodes. Each cell of the table represents a configuration of the states of the member nodes. Our program depends on this feature a lot while it is calculating the entropy of the network.
- retrieve the belief of a discrete chance node. Users can easily get the belief of a specific state of a certain node.
- save to memory functionality support efficient inference and initialization. This makes it faster to do operations in a big network.
- Junction tree navigation. JuntionTreeList is made up of junction trees, a junction tree is composed of cliques, a clique contains nodes. It is easy for users to traverse through the nodes in trees and cliques.

3. **Error handling**

   Several types of errors can occur when using a function from the HUGIN API. These errors can be a result of errors in the application program, of running out of memory, of corrupted data files, etc. whenever a Hugin Decision Engine operation fails, an error indicator is returned.

Conclusions

In this work we presented the CAT for the domain of fractions. The student model based on the Bayesian Network was developed. To build a model we had to analyze the basic operations with fractions and to determine the skills necessary for these operations. A meeting with a teacher was arranged. We have got the necessary information for building a network. We created paper tests for students to check how they solve the tasks with fractions.

We made an analysis and the results of the paper tests were used:

- to update the structure of the network, i.e. misconceptions and new skills were added to our model;
- to update the probability tables using penalized EM algorithm;
- to verify how the network fits to the particular student group.

The main result of our work is a program which implements the CAT. This program performs tests on operations with fractions. It determines what skills students have. The item bank is stored in an XML file and can be easily expanded or modified.

Our work can be extended. We concentrated on the necessary skills of basic operations with fractions and we did not deeply analyzed the misconceptions as the special test should be created for this.

Bibliography

[1] Graduate Record Examinations. Frequently Asked Questions. About the General Test. http://www.gre.org/faq.html.

[2] Fractions with Diferent Denominators. http://www.aaamath.com/fra66k-addfracud.html.

[3] Hugin Developer. http://www.hugin.com/.

[4] Java API manual for the Hugin Decision Engine V5.3. http://developer.hugin.com/documentation/API_Manuals/.

[5] Bergstron B., & Gershon R. *Computerized adaptive testing for licensure and certification*, volume CLEAR Exam Review. Winter 1999.

[6] J.Banfill. Multiplying fractions. http://www.aaamath.com/fra66m-multfract.html, 2000.

[7] Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag New York, 2001.

[8] Steffen L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191–201, 1995.

[9] R. G. Almond and R. J. Mislevy. Graphical Models and Computerized Adaptive Testing. *Applied Psychological Measurement*, 23(3):223–237, 1999.

[10] Jiri Vomlel. Evidence propagation in bayesian networks for computerized adaptive testing.

Appendix A

Tests

Appendix B

Student model

Appendix C

Results of paper tests

Appendix D

Read.me file

```
How to run a program:

This program is written in Java and uses Hugin API 5.2.
To run this program Hugin Proffesional is required to be installed on the
computer


1. If Hugin is not installed on your computer:

Download Hugin from:
  http://www.hugin.com/Products_Services/Products/Commercial/Explorer/
  a) install program

After you install Hugin Proffwsional, ensure that your Java VM can always access to:

  hapi52.jar* - This is installed in the "lib" subfolder of the
        Hugin Java API 5.2 program folder
        (eg. "C:\Program Files\Hugin\Java_API52\lib").
        Add this jar to the classpath.
        (e.g. for the JDK1.3 java VM: set the CLASSPATH environment
        variable to contain "C:\Program Files\Hugin\Java_API52\lib\hapi52.jar".

        or

        you can copy this file to folder \lib in the folder, where CATest is
        placed and modify run.bat file.(eg. if CATest is placed in folder
        C:\CATest, copy  this file to C:\CATest\lib\  and modify run.bat
        file (read instructions inside this file) )


  nhapi52.dll* is installed in the "bin" subfolder of the
        Hugin Java API 5.2 program folder
        (eg. "C:\Program Files\Hugin\Java_API52\bin"). When running the
        Java VM, this file MUST be in the search path.
```

```
        or

        you can copy this file to your bin folder of JDK. (eg. if you have
        installed jdk1.3.1 in C:\jdk1.3.1, copy this file to folder
        C:\jdk1.3.1\bin\  folder.)

*Note: last two digits of file name represents the version of program.
Our program was build using version 5.2. Your Hugin program can
have higher version and folder structure can be changed.
Try to fing hapi??.jar and nhapi??.dll files in other folders.

2. Running program

To start program type in command line:

  run.bat

 or

   run.bat /?

 to get help about key words.
```