universität wien

MASTERARBEIT

Titel der Masterarbeit

„Cognitive aspects of Designing Dialogues in

Theorem-Prover Based Mathematics Assistants

Implementation of Error-Patterns Guiding Dialogues in ISAC“

Verfasserin

Gabriella Daróczy, MSc

angestrebter akademischer Grad

Master of Science (MSc)

Wien, 2013

Studienkennzahl lt. Studienblatt:	A 066 013
Studienrichtung lt. Studienblatt:	MEi:CogSci – Middle European Interdisciplinary Master Programme in Cognitive Science
Betreuerin / Betreuer:	ao. Univ.-Prof. Dipl.-Ing. Dr. Franz-Markus Peschl

Abstract

The thesis reviews theories from cognitive science relevant for learning mathematics and applies the findings to a specific design and implementation of error patterns for calculating with fractions in the experimental ISAC system.

Since identifying the kind of design and the topic for implementation was part of the thesis, the work started from a general view: Research results on the basic structures and processes in the human brain doing and learning mathematics are reviewed; respective research results are related to the issues of abstraction and of errors in mathematics.

Particular attention is given to cognitive theories specifically developed for designing educational mathematics software; respective software products are considered in those aspects which are interesting for comparison with ISAC.

ISAC is described as a prototype for an upcoming kind of educational mathematics assistants, which are based on Computer Theorem Proving — emphasising the advantages of this technology over other systems: (1) automated derivation of user input from logical context and (2) automated proposal of next steps towards a solution in (symbolic) calculations for step-wise problem solving close to traditional calculations by pencil and paper. These services open novel chances for automated generation of dialogue guidance — demonstrating the use of these chances is the concrete point of this thesis.

The concrete design of error patterns and their implementation for fractions is described such that the description can serve as guideline for continuation of the work: The code implementing the error patterns is written in the programming language of ISAC's mathematics engine; this code comprises a few dozens of lines. The code implementing a demonstrator for user guidance is written in the rule-based language of an expert system; this code comprises four rules.

The implementation demonstrates generality and efficiency of the design: (1) the error patterns for fractions generalise to all other kinds of calculations like differentiation, equation solving, etc., (2) the implementation transfers to all usages of respective calculations in engineering and science and (3) the efficiency is show by the fact, that four rules suffice to cover all generalisation and transfer mentioned in (2) and (3).

Finally the demonstrated generality and efficiency of the error patterns gives raise to previews to the future: how these advances promise support for independent learning by trial and error also in mathematics, like simulations and games provide support in other subjects. And how such software supports renewal of learning culture.

Keywords

cognitive science, error-pattern, dialogue, feed-back, learning, fractions,
computer theorem proving, automation, rewriting, matching, Lucas interpretation.

Contents

List of Figures

0.1 Introduction

"I know that two and two make four."
George Gordon Byron

The thesis relates the possibilities of an upcoming new generation of Theory-Prover based educational mathematics assistants with the potential of cognitive science, and also to meet the challenge to design a user-friendly learning environment/dialogue system that is compatible with how the human mind actually works. The domain of mathematics is huge, as well as the domain of algebra, so we will focus on development for a concrete case: simplification of fractions, how people actually learn to use fractions, and how we can react on errors happening in this domain.

The software we use for practical work in this thesis is ISAC, an educational mathematics assistant based on Lucas-Interpretation (LIP): LIP supports step-wise solving mathematics problems in science and technology by "next-step-guidance" ,where the system can provide the next step, a novel technology developed at TU Graz and RISC Linz, indicates potential to proceed from reactive to active systems. Reactive systems are: Mechanized mathematics assistants (MMA), Computer Algebra Systems and Dynamic Geometry Systems: Where the user provides input, and the system reacts with an output. This kind of behaviour carried over to educational MMAs so far. Active systems provide more possibility for interaction with the computer. Designing the dialogue system based on cognitive aspects is a complex task, there are many questions and aspects to be taken into consideration: how to provide individual answers (in a face-to face human teaching process, the teacher is mostly able to provide individual answers for different learners), how to deal with the social aspect of learning...etc, and how to react on errors. The dialogue system will not want to solve the mathematical task in the dialogue process, but guide the learner through the learning process, with showing strategies, hints, patterns, supporting individual learning strategies, and most important: react on errors. Identifying how ISAC should react on the last is the main focus of this thesis. Source of quotes: ([107], [108], [109], [110], [111], [112]).

The first part of the thesis identifies the human way of thinking in mathematics, and tries to answer, how learning happens, and what are the innate processes in mathematics and how this could be applicable in computer based education, how could we support the learning process in the next-step-guidance system, ISAC. Understanding the thinking involved in the learning and doing mathematics is a key aspect to build educational software. The first part of the thesis also introduces the key features of ISAC. These include the first two chapters.

Cognitive science is relatively new in the field of mathematics education. It claims, that mathematics is a product of adaptive human activities,is a human-made system based on our biological and bodily experiences, and is produced by the brain abstraction, large part of mathematics are happening on the unconscious level, and mathematical ideas are quite stable over hundred years. The human brain, as also animals, has some innate arithmetic, for example comparing small numbers, and many ideas are metaphoric (Nunez [103]). Question of the thesis is how we can support the bodily based learning process, or which cognitive aspects can we take into consideration for the dialogue.

The second main part of the thesis deals with the concrete implementation of error-patterns in ISAC. This part will explain why we can use and work with pre-defined typical error-patterns. The notion of error-patterns is not new in mathematics education. Reacting on them, and providing solution strategies, hints in the right moment to solve the problem, is essential. Each learner possesses a different knowledge level, and different experiences, supporting individual way of thinking is also an important point. We try to support individual way of thinking with multiple strategies by accepting all correct answers to the step in the problem class. Secondly we try to support, that human cognitive system, and also

mathematics knowledge is fallible and learning happens partly through trials and errors, also by patterns, with responding group of error-patterns. Fractions belong to algebra, and are number made up of other numbers. Simplifying fractions might be a difficult problem for the learners.

Later, based on this work, it would be also possible to create user-models, and develop a more effective dialogue system between learner and computer. Developing and designing the concrete dialogue system in ISAC involves computer science and rule based-system, focusing on interdisciplinary aspect, connecting human cognition with concrete computer application in a real time human-computer interaction. Using a computer software in education enables the possibility to response at individual learning strategies.

Detailed structure of the thesis: As already mentioned above, the first big part of the thesis deals with general concepts about mathematics, and the domain of fractions, the second part introduces ISAC and the main task of the thesis, the third part covers the task-relevant more concrete cognitive aspects, and than go into the concrete implementation, and end with a summary. Chapter §1 reviews the state-of-the-art in mathematics education as related to cognitive science.

Chapter §2 introduces an upcoming generation of educational mathematics assistants, based on Theorem Proving (TP) technology, and a respective prototype, ISAC, selected for implementation. §2.1 introduces respective key features, §2.2 clarifies what "problem solving" means in this context, and focuses on "next step guidance (NSG)" aiming at implementation.

Chapter §3 focuses on cognitive science aspects, errors in the domain of fractions, the dialogue. It also introduces guidelines from cognitive aspects, and the error-patterns chosen in this thesis for ISAC.

Chapter §4 describes the first attempt of "dialogue authoring" in ISAC realised by this thesis. So a significant result is the manual in §4.2.3.

Chapter §5 comes back to the general aims stated in §1 and discusses how to embed the work from §4 in to learning scenarios The final section summarises interesting questions and selects a topic for practical implementation. Chapter §6 concludes with a summary and a conclusion.

Chapter 1

Mathematics Education and Cognitive Science

"As far as the laws of mathematics refer to reality,
they are not certain; and as far as they are certain,
they do not refer to reality."
Albert Einstein

The following thesis: *Error-patterns Guiding Dialogues in Theorem-Prover Based Mathematics Assistant* sets the goal to design the basis for a computer guided dialogue system for an existing educational software: The ΣAC for a concrete field: fraction, and for reacting errors in this domain. Teaching and guiding mathematics is a complex task. During a real life interaction a teacher is mostly able to react on the learner, can correct errors, may have intuitions where and what kind of a help a learner needs, including even: dealing with emotional problems, like overcoming frustration during an example. In education the most successful learning method is the one-to-one teaching, the apprenticeship like learning, but this need enormous personal costs. One way to support teaching is with the help of an emerging field of education software, because they may enable the possibility for an individual learning experience, like we would wait from a one-to-one tutoring system. According to (Nunez [101]) "primary mathematical classroom children and work modern mathematics classroom would be much better equipped to respond to children mathematical questioning and learning. But in designing the software we should be careful, because basically there are two ways: the first is to let the learners learn the software, and the interaction, kind of adopting them to the software, or to design a software that adopts to the learner. The theory of mathematical education is very broad: from sociological anthropological perspectives (cultural processes, social interaction, individual cognition, emergent mathematical environments),constructivism (conceptual schemes, abstraction, generalisation,) to cognitive theories (conceptual fields, metaphor models, development of mathematical reasoning (Cobb et al. [29])). Before going into concrete design task we have the following questions during investigating the theoretical background, and identify necessary elements to be used in ΣAC:

1. What are the cognitive processes of mathematics with the sub-questions what is mathematics? If there exists, what are the innate processes? These questions are important in the design process because we have to identify what is the subject of investigation we are extending the computer education system ΣAC.

2. How does learning happen in the domain of mathematics education, and what are the aspects we can use in an educational computer software for making the interaction more human-near.

3. Identifying the concrete elements regarding the algebra, fractions concentrating further on errors in fractions.

Summarizing, the guiding questions: we would like to get an insight how does mathematics learning and teaching works, and what can we apply in a computer software in a concrete field of the fraction? In the end we will try to merge theory with real life application, into the educational software ISAC. Of course educational software still have their limitations, so many suggestions will be included in the last part of the thesis (See Chapter §5).

1.1 Cognitive Science Meets Mathematics

"Mathematics is a language with which
God has written the universe"
Galileo Galileo

Cognitive science has rather various research focuses, and interests in mathematics education. Besides the notion of embodiment, situatedness, concept images, schemas, and metaphors according to (Fazio and Siegler [47]) Cognitive science also deals with the following topics: understanding cognition before school, the reason of pitfalls (Understanding the last would be a benefit for educational software, because they enable the possibility to collect), the discovery and insight, the way of analytical thinking, and also in the domain of learning, the conceptual, procedural, and cooperative learning, basically how people understand and learn mathematics. We will deal more with the domain of learning connected to the already mentioned sub-field, the algebra and fraction.
Before going to details lets define the notion of mathematics.

What is mathematics at all? Many would agree, that mathematics is one of the most unloved and feared subjects at school (Jennison and Beswick [62]), where most people think, mathematics needs special kind of inborn talent. For a comparison we would like to give several definition, starting with historical definitions from Greece, and ending with the definition from cognitive science. First, in the history of mathematics, it was hold to be something mysterious:

1. The notion mathematics comes from the word "Mathematikoi" (Pythagoras called his students "Mathematikoi" who studied for a longer time), and originally meant researcher (Dahl and Nordqvist [31]). Pythagorean believed that the principles of mathematics were the explanation for the whole word. Later on during the development of mathematics people came to a more realistic definition:

2. According to Oxford Dictionary (Dictionaries. [41]) mathematics is: "the abstract science of number, quantity, and space, either as abstract concepts (pure mathematics), or as applied to other disciplines such as physics and engineering (applied mathematics)", and originates from Latin (ars) mathematica 'mathematical (art)' and from Greek mathmatik (tekhn), from the base of manthanein 'learn'. Originally (Amador [2]) mathematics is an art of learning which might point out the connection to the underlying thinking processes.

3. Our last definition from a cognitive science domain according to (Nunez [102]) mathematics is about human ideas, grounded in everyday cognitive mechanisms:
"Mathematics is a product of adaptive human activities in the world shared and made meaningful language and based ultimately on biological and bodily experiences unique to our species." The concept of embodiment is supported by many researches, according to (Koestler [71]) mathematics is a conceptual metaphor system,

as also Nunez (Nunez [99]) states. (Rittle-Johnson et al. [117]) argues that "children understand and produce their embodied concepts ins mathematics, that mathematics models are embedded in the mental world".

4. We dont want to focus on understanding the thinking of mathematicians, because this might be very different from the thinking of learners(Hadamard [58]), but we would like to add still a nice definition from (Borovik [16]) about what a mathematician is: a mathematician is someone who rectifies abstract concepts intentionally and purposely and who can reuse, in compressed form, the psychological experiences of previous rectifications; the mathematician actively seeks new or known, but previously ignored, representations and interpretations of his or her objects; a mathematician has an instinctive tendency to favour objects, processes and rules with the simplest possible descriptions or formulations."

The main question was if mathematics exists outside of the world, also without human, for example in the nature, in plants, or mathematics is a human product which arises during our evolutionary development. This debate is still not closed nowadays, but in this thesis we will accept the platonic (Linnebo [80]) view of mathematics: that mathematical concepts exists in our brain. Choosing this option is important, because we would like to understand aspects from mathematics which are needed in a cognitively supportive computer learning scenario, and by a design of ISAC.

Shortly summarizing, what is mathematics from cognitive science point of view:

1. Mathematics is a language, developed by human from the centuries. (Nunez [100]) mathematics develops within a history of collective arguments and description (Lakatos [78]).
2. Mathematics is based on real bodily experiences is embodied, grounded and situated (Nunez [101]).
3. Mathematics involves the process of abstraction, and these abstract concepts are product of the human brain.
4. Arithmetic and algebra have different processes, involve different brain parts (Delazer et al. [38]).
5. Education and learning environment influences the success of education.

We have chosen fractions because they are present in other domains of mathematics: for example in engineering, or economy studies, where you have to calculate with fractions. One might understand the process of calculation but might do errors during these steps. These design presented in this thesis is exactly for these group: who once learned it (knew the basics, so the thesis will deal with the common, in the beginning usual visual representation of fractions but will operate on higher algebraic level), but for some reason they cannot properly deal with fraction, and this hinders them to solve a problem on an engineering or economic domain. In this case a personalised guidance through a software like ISAC could be very useful.

Our guiding question automatically arises: What are the features of mathematics, and what special features are important in defining the cognitively important points for the educational software, ISAC?
One, that that is surely special about mathematics is, that it is precise, consistent, stable across time and communities, understandable across culture , calculable, generalizable, and a general tool for description (Nunez [101]). Also we can say, that mathematics is the language, most people talk. To understand the important features of mathematics we should understand which processes are involved in successfully learning mathematics. For example transition/changing between domains, or according to (Borovik [16]), or dealing successfully with abstraction.

The next chapter will be guided from the following questions: What are the cognitive processes in mathematics? In details:

1. Where is mathematics in the brain?
2. Why is abstraction important in mathematics?
3. What are errors in mathematics?
4. What is special about algebra, and fractions?
5. What is the role of a learning environment?

1.2 Mathematics and the Brain

"Mathematician work where
demons lurk-deep in the interior of the mind"
Keith Devlin

If we would like to understand mathematical thinking, the first question is, whether mathematics has inborn capacities, and if there exists innate processes. The question is important, because if mathematics is mostly inborn, it is "pre-determined" who is better in learning mathematics, and who not. If we are able to define the parts of mathematical capabilities and skills which arise during the learning process we can work on them to build support software, or create learning environment which helps in the learning process. There is also the philosophical question we already dealt with in the first chapter whether mathematics is created by the human brain or mathematics exists outside of our cognition. However, there are people who believe in the "universality" of mathematics and the latter view was a leading view among mathematicians, and other people, mathematics is a production of the brain, there is no mystery: "Mathematics is produced from our brain" (Borovik [16], Nunez [101], Devlin [40]), "and depends on the historical development" (as mentioned above).

As we could see, by the definition of cognitive science, mathematics is created by human mind, and mathematics conceptual are constructed as a common set of neural and bodily structures, and mathematical theories are grounded in experience of embodied ideas (as defined above). Especially arithmetic is based on physical experiences, space size and motion. Of course how is also a still open research question. Very interesting is the book from the mathematician: (Borovik [16]) Mathematics under the microscope , who also states that we cannot understand the nature of mathematics without understanding first the interaction between learned and/or invented mathematical processes and underlying powerful built- in, inborn algorithms of our brain, and we have to take care at the individual cognition, if we want to understand the process of mathematics, the patterns of the brain".

Our key focus of this section is to find out if algebra is somewhere hard-wired in the brain or not. According to (of Teachers of Mathematics and Mathematical Sciences Education Board [104]), algebra is more, than fluency in manipulating symbols, It involves representing analysing and generalizing patterns, using tables, graphs, words, and symbolic rules relating and comparing different representation. But is it inborn?

Because mathematical ideas are quite stable, we could think that most mathematical ideas are innate from nature, and the question is do we share things with animals?

1. Rhesus monkeys capabilities are similar to the infants. Babies, chimpanzees share many abilities: Babies with seven month were able to decide number equivalence between arrays of objects. Chimpanzees can do simple mathematics: They understand

some simple fractions, like $\frac{1}{4}$, $\frac{3}{4}$, $\frac{1}{2}$ and are also able to tough to learn calculate using numerical symbols (Lakoff and Nunez [79]).

2. Every innate process seems to be connected to the so called number sense, that term was introduced by (Danzig [33]) that is that "the brain is born with a very basic sense of quantity-this is what we share we animals- and the role is to subitizing very small objects".

Other conclusions are: there is no innate number line (Nunez [103]), or higher mathematics in the brain, rather some basic ability to compare, judge numbers, estimate the number of objects in a group, order, pair, memory, exhaustion detection, and pattern recognition. And we cant forget, that mathematical concepts are attached to minimal innate arithmetic, and only small numbers could be hard-wired.
This answers question of basic levels, but not the level mathematics education takes place. Higher mathematical thinking, is still an open question: It seems there is no inborn higher mathematics, usually it is learned through the years. So the way of teaching is a key in higher mathematical thinking processes, and its development is quite long for such an abstract notions like 0, -, imaginary numbers.

It is also very interesting which brain parts are involved in mathematical processes. For this question we can shortly examine research results from fMRI studies.

1. Pre-frontal cortex is involved in the complex structuring used in arithmetic calculation.(Blair et al. [12]): inferior parietal cortex is involved in symbolic numerical abilities. This part is responsible for the knowledge of number sequence, and is highly associated, connected with audition, vision, and touch (Grabner et al. [56]). Memorization, and multiplication tables are associated with basal ganglia, memory, pattern. (Kucian et al. [75])

2. "Neuroimaging studies show a link between mental calculation and the angular gyru, which is to be hypothesised to be related to arithmetical abilities which also depends on the level of training" (Grabner et al. [55]).

3. Other fMRI studies (Cantlon et al. [23]) has shown, that " Adult humans, infants, pre-school children, and non-human animals appear to share a system of approximate numerical processing for non-symbolic stimuli such as arrays of dots or sequences of tones. and their results support the claims that there is a neurophysiological link between non-symbolic and symbolic numerical processing in adulthood." But mathematical processes are also culture dependant (Cantlon et al. [23]).

4. However, there are studies connecting brain parts to numerical abilities: For example according to (Grabner et al. [55]): "Functional neuroimaging studies have revealed that parietal brain circuits sub-serve arithmetic problem solving and that their recruitment dynamically changes as a function of training and development." That means, that even if we have inborn mathematical capabilities, it might be changed through training.

5. The role of short and long-term memory (Borovik [16]) is also an important question. Thus we would like to decrease the cognitive load..etc. Cognitive load is important as human we are limited in our observation. Very important is, that algebraic calculation, and arithmetic seem to be very different, and do belong to different brain regions. According to (Devlin [40]) algebraic calculation and arithmetic are in different brain region, and also other talk about the cognitive gap between arithmetic and algebra.

What connects mathematics to other thinking processes? Mathematics might also involve cognitive processes, which we not only use in mathematics, but also in other domains: "How much of mathematical understanding makes use of the same kind of conceptual mechanism that are used in the understanding of ordinary non-mathematical domains (Nunez [99])? That raises the question, that which domains we have to take into consideration during planing the software. Since mathematics is embodied, and mainly unconscious, the ability to adapt other cognitive mechanism for mathematical purposes is also part of higher mathematical thinking, that according to (Graber et al. [54]), because the neural motor-control program has the same structure. For example to "create mental imaginary without visual input" (Mitchelmore and White [88]) fundamental mathematical ideas are closely related to the real world and their learning involves empirical concepts.

Our conclusion for design decisions However, current neuroscience studies cannot provide enough information how the higher cognition of mathematics work, and what we can use in a practice, but these studies shows, that mathematical abilities depend on training and education. There are some innate processes in the field of arithmetic (Nunez [99]), but algebra is obviously a learned process, in those, being possible to support this learning process through a software. Basic number sense also differentiates from higher mathematical thinking, and mathematics is connected to other domains. Higher mathematical thinking is strongly involved in solving fraction, and algebra problems, so in the next chapter we will focus on the abstraction, and exploring patterns.

1.3 Abstraction and Algebra

"I have created a new universe from nothing"
Janos Bolyai

We have to investigate the question of abstraction, because it is directly or indirectly connected to fraction. (Booker [15]) states: "Fractions are the first abstracted mathematics met by the young learners" . It is usually said, that mathematical concepts are abstract concepts. In the process of mathematical abstraction (the process includes generalisation) the dependence on real world objects will be removed.

Abstraction is a key concept in mathematical thinking, because it helps to reveal connection between different areas of mathematics, a known results in one area can suggest conjecture in a related area, and techniques and methods from one area can be applied to prove results in a related area. (Mitchelmore and White [89]) argues, that an abstract mathematical object takes its meaning only from the system within which is defined, because every single world, syntax is defined precisely, and even if mathematics uses everyday words, their meaning is not their everyday meaning in mathematics. Investigating the question of abstraction would bring up arguments against embodiment: according to (Reif [113]) scientific and mathematical concepts are significantly different from everyday concepts. As we have seen above, that mathematicians dont think the same way as learners of mathematics do, we have to raise the following question: What does abstraction mean, is the above mentioned mathematical definition the valid one also in educational settings, and learning scenarios? As in many fields, this concept has also different interpretation, and is defined in various fields differently.

Definition of abstraction in teaching and learning: Even if it seems that abstraction in mathematics is separated from real world objects, however if we think of it, the notion of abstraction originates from Latin: past participle of *abstrahere*, meaning to draw away. If we think, drawing away still means a connection from the original concept. According to

(Mitchelmore and White [89], and (Koedinger et al. [70]), in learning processes, mathematical abstraction are different. (Mitchelmore and White [89]) defined the notion of empirical abstraction, which later will became a mathematical abstraction with linking mathematical objects to empirical objects. According to (Skemp [120]) in empirical abstraction is: "Abstracting is an activity by which we become aware of similarities ... among our experiences". (Nunez [101]) states that embodied mathematics also involve generalization over poly-semi, influence of patterns, new examples of conventional mapping, conceptual blending: (conceptual combination of two distinct cognitive structure with fixed correspondence between them, and symbol associated with the concept. We mentioned these things in the beginning of this chapter, because, these are all connected somehow to abstraction processes, and the thesis deals with a section of algebra, these are the domain of fractions. How does this abstraction working? Maybe that is the most open question in mathematics at all. (Nunez [101]): "Mathematics is a highly technical domain characterized y the fact, that the very entities that constitute is are idealized mental abstraction. These entities cannot be perceived directly through senses. Even the simplest entity in say Euclidean geometry cannot actually be perceived. I will build on the increasing evidence showing the extremely close relationships between speech, thoughts, and gesture production at a behavioural developmental psychological an cognitive linguistic level."

Process of abstraction and cognitive skills required to solve algebra problems The problem with abstraction is, that some parts seem to be not easy to learn. Part of the abstraction seems to be there from a quite early age (Koedinger et al. [70]). (Warren and Cooper [135]) states, that formal algebraic notion are not easy to learn.

One important part in the abstraction process might definitely be the patterns. There are many patterns in mathematics: like binomial forms, or numbers are very pattern based. Learning to see this relationships between mathematical objects, the transformations, grouping pattern, generalization is very important in functional thinking. This functional thinking is present from very early on. Also young children are capable for thinking functionally (Cheng [25]). Part of understanding the process of mathematical learning is to understand the thinking involved in doing and learning mathematics.

Besides being able to make abstractions: as mentioned before it is already clear, that algebra requires completely different abilities, and understanding, and teaching methods as arithmetic. Algebra is very interesting, because being good at algebra requires different cognitive abilities, than arithmetic.For example, the cognitive abilities for arithmetic are the following: *Grouping ordering, paring, memory capacity, exhaustion capacity*, which are present in the process of abstraction.

Not only the process of abstraction, but also the transition between different domains is very important. (However, it is out of scope of the thesis, in an ideal case a software would also provide help to support the process of transition, so we found important to show some existing theories on it.)

"Cryptomorhism" or "transition between domains" We can also see mathematics as system of abstraction and representation, and transitions between these representations. Compared to other abstraction systems mathematics and science are more rigid, and fix. Still the concepts and symbols are quite easy to change. Like a may be replaced by b, c, or any other kind of symbols, not always the name is important, but the features, and the rules of the current representation system. According to (Michene [86]) understanding mathematics also contains creating associations of many kinds, and differentiation between various kinds of items. Transition between domains is strongly related to multiple representations are also said to support cognitive processes in learning (Ainsworth [1]). Multiple representations help also to overcome the problem of the transition between domain, also helping the constructivist approach to education. Other goals would be to "promote abstraction, to encourage generalisation and to teach the relation between representation"

(Ainsworth [1]), because one single mathematical object can have multiple readings.

This field is not well researched, but this kind of changes, or connections to other domains are called ”crypthomorphism” (Borovik [16]), or ”transition between domains”. This means switching between two representation, and with these also realizing what are the similarities, and differences between the two representation, this is something we do since we are small kids, as well as described by (Borovik [16]), this domain needs further investigation, and clarifying basic notions. Transition of domains may pay an important role in transforming knowledge, and changing strategies, creating solutions (Rittle-Johnson and Siegler [116]), that is why we definitely need to focus on it, because this is strongly involved in the learning process.

Transition of domain occurs quite early in other form of thinking processes too, like analogies: (Clement [28]) dealt with the research question: where do spontaneous analogies come from? ”An analogy involves a shift in the problem representation. ”However, it is a shift of a special kind-this is a horizontal change, and a vertical change is if to move to a more abstract representation.” Using an analogy is the most creative of these three strategies in the sense that one is shifting ones attention to a different problem, not just to an abstract version of the same problem ”features ordinarily assumed fixed in the original situation are different”.

Does technology support ”cryptomorphism”? (Ainsworth [1]) states that technologies are also able to support multiple representations, and ”it should be supported to maximise learning outcomes and for using more than one representation is that this is more likely to capture a learners interest and in so doing, play an important role in promoting conditions for effective learning”. In *ISAC* different representations can be supported in the *html* content. Also in educational software (Brown and Burton [19]) transition between domains were taken into consideration regarding the errors: ”lack of transfer is often referred to as shallow learning, that occurs in many form of instruction and in many domains.”

Conclusion of the chapter: Mathematics works on the level of the abstraction (object is present, object known but present, learned imagined object, or there is no simple link to the world), on the last level. Research on the learning of algebra is still in its infancy since we have not yet been able to see what the long-term effects of different computer-supported interventions are, the findings presented in this section should be taken primarily as indications of areas in algebra where the use of computer has already yielded interesting results and where ”further research is likely to be even more fruitful” (Kieran [66]): to build a good educational software, we have to pay attention both to the differences between arithmetic and algebra and the process of abstraction till it is key in learning, discovery in science, process (Clancey [27]). According to (Mitchelmore and White [89]) : Abstract general embody general properties of the real world. Different kind of mathematical mind, meta-theoretical correlation, concept image and concept(Dreyfus [44]).

However, it is not a question in this thesis how learners develop and conceptualize these representations, rather the goal is to develop an interactive dialogue between a learner and software (Ritter et al. [115]).definition.

1.4 Fractions

"Either mathematics is too big for the human mind
or the human mind is more than a machine. "
Kurt Godel

The aim of this sub-chapter is to focus closer to the domain of fractions, the typical errors occurring during problem solving by this domain, why learners dont like fraction. Without loosing focus: with the help of these findings we work towards defining the cognitive elements important in the design question for this specific domain. For this domain of mathematics described in this thesis there was, in principle partly, free choice for all domains of mathematics, but in fact limited to the domains already implemented in the ISAC prototype. Learning in this domain causes well-researched difficulties (Kieran [66]), and is most unloved subject in mathematics education among learners.

What are fraction, and why are they so scary? Complex problems of fractions are fascinating, because they involve many skills and knowledge previously learned: learners have to be able to carry out (in right time, and order) almost all previously learned mathematical examples and knowledge. Examples in this domain vary from simple to difficult. For example simplifying the following fraction: $\boxed{\frac{5a}{a}}$ is rather simple, and simplifying the following one: $\boxed{\frac{12y}{x-5} + \frac{2y}{x+3} + \frac{16y}{x^2-2x-15}}$ is rather difficult, and thus involves also for example addition, factorisation of terms.

As fractions are also part of algebra, there are many rules one should take into consideration by solving an example (Associativity, commutativity, distributivity, and there are several, sometimes even different rules for the operations too, e.g.: nominator and denominator dont behave the same way).

Definition of Fractions:

1. According to Oxford Dictionary (Dictionaries [43]) fractions are: "a numerical quantity that is not a whole number (e.g. 1/ 2, 0.5), basically they are mathematical objects, which also represent a relationship between two numbers".
2. According to (Nunez [100]) the embodied view fractions are number made up of other numbers, a part of a unity object made by splitting a unit object into n parts. We can split up to simple fraction 1/n, a complex fraction: n/m. From embodied point of view we can do some action with these objects. The action for the fractions can be following: to divide, to bind the parts, to measure, and to do operations. However, the above represented example which also included addition surely involves also other concepts.

What can we do with fraction, how do they look like, and what do they represent? Fractions are especially hard, because their domain summons and requires many different mathematical skills: as the learners has to be able to understand the difference between variable, and numbers, to be able to correctly operate on single elements and on whole fraction, to recognize well known, and previously learned patterns: binomial formats.

Typical operations usually carried out with fractions are the following, and are represented in the correct way (no errors here!):

Division: $\boxed{\frac{a}{b} : c = \frac{a}{b \cdot c}}$ or $\boxed{\frac{a}{b} : \frac{c}{d} = \frac{a \cdot d}{b \cdot c}}$
Multiplication: $\boxed{\frac{a}{b} \cdot c = \frac{a \cdot c}{b}}$ or $\boxed{\frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}}$

Addition or subtraction: $\boxed{\frac{a}{b} + -c = \frac{a+b\cdot c}{b}}$ or $\boxed{\frac{a}{b} + -\frac{c}{d} = \frac{a\cdot d+c\cdot b}{b\cdot d}}$ or $\boxed{\frac{a}{b} + -\frac{c}{b} = \frac{a+-c}{b}}$

Extension: $\boxed{\frac{a}{b} = \frac{a\cdot c}{b\cdot c}}$

Fractions **vary the form**, e.g. from a mixed form $2\frac{3}{5}$ to $\frac{13}{5}$ or 2.6, (see also figure, where each fraction is the same and represents the same value)

$$\boxed{2\tfrac{3}{5} = 2 + \tfrac{3}{5} = \tfrac{13}{5} = 2,6 = 2\tfrac{3}{5} + 0 = 2\tfrac{3+0}{5} = \tfrac{13}{5+4-4} = \sqrt{(2\tfrac{3}{5})^2} = \tfrac{130}{50} = \tfrac{13\cdot 4}{5\cdot 4}}$$

Fractions **represent**, and establish relationship between two numbers, involve multiple representation:

1. Division $\boxed{\frac{13}{5} = 13/5}$,
2. proportion $\boxed{13{:}5 = a{:}b = 26 : 10}$,
3. Linear change $\boxed{\delta y{=}13\ \delta x{=}5}$,
4. Simple value $\boxed{2{,}6}$, etc.
5. Relation to real world objects: fractions are thought (especially in the beginning): like a piece of cake, a pizza (as shown in in Fig.5 on p.16. Thus this is a very common method in teaching fractions, and we can consider this as an embodied aspect, we have chosen in this thesis to concentrate on higher processes, and assume that students already have these real-world connections.

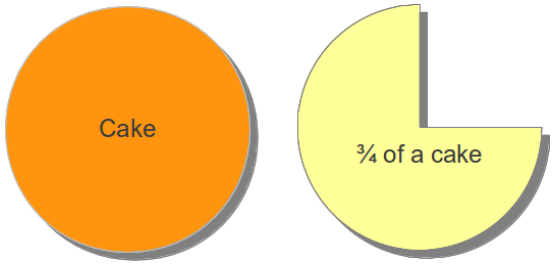

Figure 1.1: Fractions Represented as a Piece of Cake.

(Carraher [24]) states: "the field of fraction still lacks a coherent account of how fractions fit into learners mathematical understanding, beginning from early childhood and extending into late adolescence, and how this are linked to representations, schemes, ideas, and concepts as decimal numbers (e.g.: decimal numbers)." For example: most learner do not think that 3:4 expresses the fraction 3/4!

If we investigate fractions, we also have to deal with the so called special numbers, as $\boxed{1}$, and $\boxed{0}$. The first represents the whole, the second something which exists, but in the same time does not exists: empty, nothing, destruction, low. (We should pay attention, that understanding special numbers is very intuitive (Lakoff and Nunez [79])

Emotional aspects and fractions As mentioned above, fractions are one of the most "hated" mathematical domains among learners. Generally we can say, the process of learning mathematics is often bound to emotions like anxiety and frustration (Hembree [59]), and also reported by (Jennison and Beswick [62]), when learners conduct error, and this frustration decreases motivation of learning mathematics a lot, "missing knowledge in a

single domain influences overall mathematical skills, and learners lack of competence with fractions is a major influence on their overall mathematics competence".
These emotion may arise due to several reasons: because of teachers, parents, teaching according to (Turner et al. [129]) or learners lack of competence in skills, or simply due to individual differences, till not everybody is learning mathematical concepts at the same speed in schools.
How to deal with emotions in a computer based learning environment is out of question of the current thesis, till they are severe enough, to be mentioned, and to be taken into consideration during the design process. As a benefit, computer software might also decrease the anxiety bound for example to persons, and provide a platform where learners might experiment freely with mathematics in their own rhythm, without pressure . However in a mixed learning environment, where teacher is also present, it would be important to deal with this question separately.
An other way of overcoming frustration is to gain confidence and filling up the gaps by the domain of the fractions.

1.5 Mathematical Errors

"I could never have gone far in any science
because on the path of every science
the lion Mathematics lies in wait for you."
C.S. Lewis

Errors in mathematics, and its domain algebra have been in the center of investigation quite a long time ago (Buckingham [22]),and were theorized due to several possible factors. One of an earlier example according to (Young and OShea [140]) subtraction are "due to the use of incorrect strategies rather than to the recall of number facts". This can be interesting and important for us, because the domain of subtraction can be also found in fractions, since they involve several domains: subtraction, arithmetic, and variable, and investigating this question could also answer, what can we do in the case learners conduct an error, and how to answer in this case.

Our guiding questions of this section are:

1. What causes errors?
2. Are there typical errors, or patterns of errors?
3. Error-patterns in algebra and fractions?

What causes errors? Typical errors are called in the didactics of mathematics misconceptions, a mistaken idea, a misunderstanding, which notion would suggest from cognitive science point of view, that we knew what a learner does not understand. This is clearly in contradiction with the findings out of how mathematical thinking works.
However in the field of mathematics didactics lot of topics deal with misconceptions, it might be, that these originate from a very natural way of human thinking, and are necessary part of the learning process, as also (Swan [125]) states: "misconception is not wrong thinking..., it may in fact be a natural stage of development". According to the empirical study of elementary errors from (Payne and Squibb [105]): "Important insight into the nature of cognitive skills and its acquisition can be gained by examining errors. "
And we have to take into consideration that trials and errors might also be part of the learning process. That means, we should leave a free place to try out things, and to make errors

during solving the mathematical tasks. However, that might be also other processes involved, not only the process of trial and error. According to (Sloman [123]) "human are born with a sophisticated, evolved, acknowledgement about how to learn in a complex three dimensional world containing more or less complex structures and processes, opportunities and obstacles, and also other intelligent individuals". He also argues, that human possess a biologically useful productive laziness (mathematicians consider themselves very often "lazy", and explain why they are looking for an easy, elegant solution, or generalisation) which reduced or eliminates the need for empirical trial and error, avoiding empirical experiments that could be dangerous or even fatal. However, the mechanism are not infallible, testing and debugging may be required. Even if errors are natural part of learning process or not, this last statement also supports the need for allowing making errors.

Errors due to communication Mathematical objects undergo when transmitted between people and institution. Many errors happen due to communication problems, or are realised as errors, since understanding what the other (in this case learner) understand in mathematics is also a social process. Mathematics can be seen as a socially shared and logically structured conceptual system (Godino [51]). So it might happen that the problems are not in the heads of the learner, but due to the communicational, ontological problems. Mathematics is transmitted via natural language, and additionally via a very strict mathematical formal language. And being able to manipulate the mathematical language might take years, so even if a younger learner would think of a good concept, but would not be able to communicate it correctly.

Communication in ISAC is limited compared to human conversation. We use text, and visual inputs, but no voice, or gestures.

Where do these errors originate? Where these errors originate is a really hard question: according (Brown and VanLehn [20]) "there is no simple one to one relationship between the types of errors and the type of faults. One has to look the relationships between underlying faults and the entire collection of errors a child makes on a set of problem". The question is complex, because: Even if one has the right idea, one can still miscalculate. Two issues in contemporary research into the teaching of mathematics are that learners can sometimes obtain correct answer by the use of principled but incorrect strategies, and their adaptation of correct but untaught alternative methods. This is also valid vice versa: Also can get incorrect answer by the use of correct strategies.
There are many possibilities why error occur: E.g.: Not knowing the rules. Due to thinking of a different thing, different representation, and actually the transition between domains is not working, due to the process of learning, or simple due to the communication between teacher and learners. By learning new concepts, learners usually go through a repetitive process, which contains different stages: first the learner gets familiar with some domain, notifies certain generalisation, and other stages may also involve, diverting, debugging, deploying the knowledge. Errors may happen on any of these learning stages.

Do this errors happen randomly? The question is important, because if errors happen only randomly, and there is no system behind the whole procedure, we should collect thousands of possible errors, and it would make not much sense to focus on errors in the first place in designing guidance. But according to (Brown and Burton [19]) "Errors are made because certain component is missing" and found to be quite pattern like, and depend on complex factors: procedural knowledge, incorrect strategies, mental representation of variables, explanations of teachers, learning environment..etc.

Errors in mathematics (we will investigate this question in the next section of these chapter) are thought to be due to misconceptions "An essential point would be to help learners to identify the sources of their misconceptions and their systematic procedural

errors. This thesis will not deal with the sources of the misconceptions, rather we will focus on these pattern like non-random errors which happen during learning process, are not random, and happen usually and we will talk about ”error-patterns”, and we will use this notion to describe errors, . error-patterns are no knew concepts (Kim and Kang [67]) collected error-patterns in division and fractions, by elementary learners, and also (Brown and VanLehn [20], Young and OShea [140]) dealt with the trouble with error-patterns of children.

1.6 Computer in Education and as Learning Environments

”We must regard classical mathematics as a combinatorial game played with symbols.”
John Neumann

So as we have seen in above chapters the ideal case would be to understand the thinking involved in the learning and doing mathematics and to build an adequate software: To support learning and teaching. Some people argue on the effective of education software, the usage of any kind of digital, and new media is increasing, and very helpful. The use of computer software is natural in the daily part of their life for some generations.

However, using computer in education is a developing area. New media may provide the possibility to change education in a way that fits human cognition better, and focus on individual aspects. Taking individual aspects into consideration during the teaching process is very hard to realise in a classroom from a resource point of view, simply because the number of teachers is limited. (Balacheff [7]) states that ”Cognitive processes need certain tools for their adequate execution. Such tools are, for example language, graphical representations, concepts and prototypic representatives, symbols and symbolic operators, mental images, and models of material situations. The computer appears able to offer quantitative new thinkings tools... Such a system should involve a learner model. This means that in the course of the teaching and tutoring process the computer tutor constructs a model of the cognitive structures of the learner from the learners reactions as far as they pertain to the intended learning process.”

In these thesis these responses should be based on interaction answering the questions: How should ISAC respond if errors happen?

If we want to plan guidance for ISAC we should investigate generally the question: Where does learning takes place? Can we see the computer, as a learning environment, or if we plan to use computer as a tool in the classroom, how can it be used ideally?
According to (Nunez [99]), learning takes place within embedding social context that influence the whole learning process. Can we see computer also as social context? To see them as a social context, and not only a passive place, computers should give first (adequate) response. We can also see computer environments as ”microwords”. The notion comes from (Edwards [45]) and ”can be said to embody mathematical or scientific ideas addressed.”

Environment in learning plays a significant role: experiments with children (Taylor and Schwartz [127]) have shown, that the physical environment can support the development of fraction concepts. The above mentioned experiment has also shown, that ”children who learned by adapting relatively unstructured environment transferred to new materials better than children who learned with ”well-structured environment”. That supports one one hand the concept, and importance of embodiment in mathematical teaching, on the other hand it supports, that to design an educational software, we should be relatively domain independent. Learning environment is also very important because mathematical

objects undergo a change when transmitted between people and institution (Godino [51]).

Learning environment also depend on the actual curricula offered in the given country, or classroom: learning environments, and the ways of teaching mathematics changed through the years according to (Blair et al. [12]) in the US, "At the turn of the 20th century, much of the mathematics instruction for children in the upper elementary grades was rigid, formalistic, and emphasized drill and rote memorization. In the 1980s and 1990s gave rise to dramatically different kinds of texts that are increasingly complex, comprehensive, and cognitively challenging". Also technologies created a new perspective on learning environments, which includes social environment, since teaching and learning situation can be considered as a whole (Bottino and Chiappini [17]).

ISAC **is planned for various environments** . On one hand for individual learning, and also for classroom teaching (even for collaboration), where a teacher is present. As learning environment we define for ISAC the following: a *social* broader environment, the *teacher* who guides the learners, and also the *individual* or learner with his/her own cognitive abilities:

1. **Social environment**:
   Mathematics was developed during long centuries, and social interaction played a key role. Human beings are able to construct mutual understanding through social interaction. But also an open question, does a child need an environment to come up with a mathematical knowledge (Sloman [123])? Mathematics has cultural and historical dimensions, so we might understand social environment. This includes: Situated learning, community of practices (situated learning, learning skill). In social learning, there is also the notion of scaffolding that provides support the learner as they carry out different activity, and it gives advice when the learner does not know what to do or is confused, guides tours on how to do things, hints when needed. Reflecting on the learning is very important in order to look back on their performance in a all situation, and compare to others. (Fleener [48]) et al argued for the importance of social groups, learning circles. Computer programs can provide a micro world of representation, supporting so the idea of embodiment (Edwards [45]).

2. **Teacher**:
   Teacher are also very important part of the social environment - Teacher is important because everybody has his/her own conceptual system, that they transform for the learners (Sloman [123]). Learners are influenced of the teachers external representation, in both procedural and conceptual knowledge (Bills [11]). That might be a reason why some learners might have problems with mathematics. According to (De Corte [37] et al important is the teacher-learning environment.

3. **Individuality**:
   As we mentioned that mathematics roots in embodied concepts, till everybody thinks individually, one can have totally different induction about the notion of infinity for example, or about paradoxes of mathematics: like how can be something at the same time limited and infinite, however, what are the key features explaining the stability of mathematical ideas. "At certain point, each children express their individuality. " However, intuitiveness is not subjective, but might result from patterns (Nunez [102]). But we mentioned that letting learners choose their individual strategies helps them to get better. Software and their individual dialogue systems open the possibility to individual learning, and to pay attention at the differences. (Koedinger et al. [70]) supports computer as an environment for exploration of fundamental ideas. *ISAC collects the learner's steps in problem solving, and this thesis shall use them to promote learning as described above.*

Short summary: (Kok [72]) discussed the importance of learning environments designed on cognitive perspectives, and there is an ongoing change in education. Important is also to design a learning environment, which is cognitively demanding, and also engaging in the same time. ”both human cognition and technology have their own weaknesses and strengths the key to constructing the most efficient systems would be through understanding the characteristics of human cognition and technology and then integrate their advantages.” One possibility is to create so called mixed reality, or blended learning environment, what the teachers can use as support tool at learning and school, and at guiding cognitive.

Chapter 2

TP-Based Mathematics Assistants Introduction to ISAC

"the mathematical sciences particularly exhibit order, symmetry, and limitation; and these are the greatest forms of the beautiful."
Aristotle

This chapter provides the prerequisites for the major part in the thesis, an implementation of error-patterns in a novel type of mathematics assistants. The implementation concerns a few lines of highly abstract code written in the "Standard Meta Language (SML)" (Milner et al. [87]) for computer mathematics, four (indeed, only four) rules for the dialogue component of ISAC, one of these novel mathematics assistants — and creates a significant high impact on ISAC's dialogue guidance.

The previous section already stated some high level requirements on tutoring by software in mathematics education, and requirements planned for ISAC. In this section reasons need to be presented, which make approaches more promising than traditional types of educational software when fulfilling such requirements.

The significant reason for more promising approaches is a technological advancement: while up to now the most powerful educational math tutors are based on Computer Algebra Systems [1], whereas ISAC is based on technology provided by the Computer Theorem Prover (TP) Isabelle (Nipkow et al. [98]).

ISAC is an experimental system, initiated at Graz University of Technology in 2002 and presently developed at the Institute for Software Technology[2], the Institute for Information Systems and Computer Media[3], as well as at the Research Institute for Symbolic Computation (RISC)[4] at the University of Linz.

Before going into technical details of ISAC the principal advances of TP technology for educational mathematics assistants are discussed.

[1] With only two exceptions, (Back [6]) and (Melis and Siekmann [85]).
[2] http://www.ist.tugraz.at
[3] http://www.iicm.tugraz.at
[4] http://www.risc.jku.at

2.1 Features of Software Based on Theorem-Proving (TP)

"If only I had the theorems!
Then I should find the proofs easily enough."
Georg Bernhard Riemann

The discipline of (Computer) Theorem Proving is as old as the discipline of Computer Algebra. While the latter provides systems indispensable in practice of engineering and of science (e.g. Maple(Meade [84]) or Mathematica(Wolfram [138]) for decades, TP only recently comes to the foreground (e.g. Coq(development team [39]) or Isabelle(Nipkow et al. [98]) with systems indispensable for coping with increasing complexity and dependability of software: in the same way other engineering disciplines evolved to sciences by strengthening mathematical foundations and methods, software engineering evolves to a science with TP as fundamental tools.

The ISAC-system does *not* use Isabelle as is, but uses specific components of Isabelle and assembles them to a system with novel features, which might be considered constitutive for a new generation of mathematics assistants (Neuper [94, 96]). Such features are:

1. *Check user input automatically, flexibly and reliably:* Input establishes a proof situation (for *automated* proving) with respect to the logical context provided by TP. Without TP technology the check for each step required separate code, which cannot be accomplished for all the variants desirable in problem solving. Thus stepwise problem solving, the predominant activity in contemporary math classes, is *not* covered by contemporary educational software (with two exceptions, (Back [6] and (Melis and Siekmann [85]) mentioned above).

2. *Give explanations on request by learners:* Due to the "LCF-paradigm" (Gordon et al. [53]) (almost) all mathematics knowledge underlying a TP system is implemented in predicate calculus (and not in some programming language), i.e. in traditional mathematical notation. So these sources of knowledge are human readable and thus are ready for inquiry by learners[5]. The challenge for development of educational software is not anymore, to create such knowledge, but just to filter out specific parts of the knowledge relevant for specific learning situations.

3. *Propose a next step if learners get stuck:* This feature is a most recent achievement in combining deductive technologies from TP with computational technologies from programming languages. Lucas-Interpretation (Neuper [95]) provides so-called "next-step-guidance", where a program "knows" an algorithm solving a problem at hand, and from that algorithm calculates a next step towards the solution of a problem. The technological challenge is to combine such steps with maintaining the logical context, which preserves feature no.1 above.

Fig.2.1 represents ISAC's reaction on a correct (good) and on a incorrect (bad) entry.

These three features of TP technology give raise for a completely new role of educational mathematics software: The traditional roles have been previously discussed, the new role is: being an interactive and transparent *model of mathematics* itself (and not being specific tools for particular tasks), reflecting essential features of mathematics such, that learners can learn by watching the model at work, by interacting with it and obtaining feed-back from "trial and error" — similar to the kind or learning chess which is featured by good chess software. TP-based software might be a model of mathematics in the same way, as chess software might be considered an interactive model of chess. However, the latter is not "transparent" in the sense of pt.2 above: there is nothing to learn from the (program) code, which would be relevant for playing chess.

[5] Isabelle's knowledge can be reviewed at `http://isabelle.in.tum.de/dist/library/HOL/index.html`.

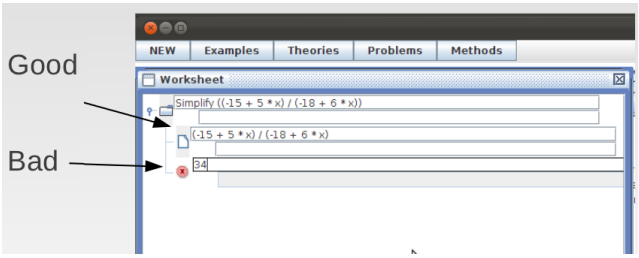

Figure 2.1: Reaction on Good and Bad Entry in ISAC

The three features are exploited by ISAC's services, which provide technological prerequisites for implementation of error-patterns in this thesis as follows:

1. *Check user input automatically, flexibly and reliably:* This service allows to detect error-patterns within arbitrary steps in problem solving. This service also allows to check input formulas within the dialogue developed in this thesis. This specific usage of the service actually involves only basic mechanisms of TP technology (typed matching and first order unification).

2. *Give explanations on request by learners:* This service, derived from the human readable mathematics knowledge, is not used by the error-patterns as implemented in this case study. How this feature can be used in further developments, this is addressed in §5 and §6.

3. *Propose a next step if learners get stuck:* This service is used to compute the correct formula in case an error-pattern has been identified. The correct formula then is presented to the learner with specific gaps to be filled in (see "fill-forms" in §4.5); respective feedback is again created automatically by the above service no.1.

Fig.2.2 shows several possibilities for interaction with ISAC

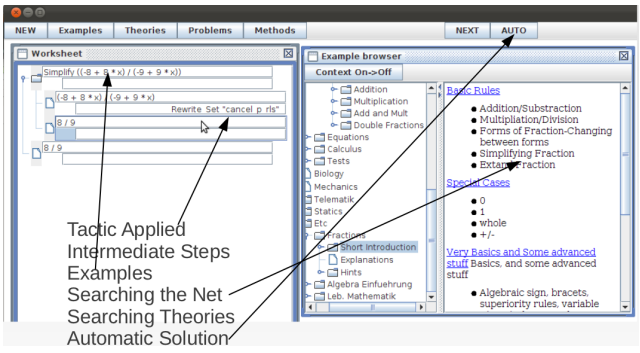

Figure 2.2: Dialogue with ISAC

Since TP technology is highly complex, ISAC is designed such that the tasks of "dialogue authors" is strictly separated from the tasks of "mathematics knowledge authors": the latter are not concerned with dialogues at all, not even with input/output of the system (Ročnik [118]). On the other side, dialogue authors are concerned with input/output, feedback and user guidance only — they only need intuitive understanding of the TP features above, they are provided a fairly simple interface, which hides the technicalities of TP and allows to focus on the comprehensive task of dialogue design.

2.2 Stepwise Problem-Solving Modelled in Software

"There is no one giant step that does it.
It's a lot of little steps."
Peter A. Cohen

As mentioned above, "step-wise problem-solving" can expect specific and novel support from TP technology — and this kind of activity is predominant in teaching mathematics: math classes at high-school, academic courses on applied mathematics in science, technology and engineering (STEM)[6] exercise "step-wise problem-solving" in order to demonstrate the usefulness of theory, in order to transfer theoretical insight to practical problem solving, in order to prepare learners for practice in engineering and science, etc. This is an example problem from a textbook in structural engineering:

> *Determine the bending line of a beam of length L, which consists of homogeneous material, which is clamped on one side and which is under constant line load q_0; see the Figure below.*

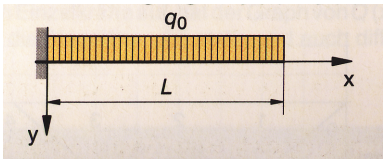

Figure 2.3: Balcony Under Load.

Scientific methodology in applied mathematics separates the problem solving process into these (at least) three phases:

1. **Modeling** transfers structures from the "real world" to the world of mathematical formulas, develops bodies of formalised knowledge and proves properties of relations between knowledge items.

   In the example above such knowledge items are "beam" or "constant line load". Relation between such items are described by mathematical theories (or mathematical theories applied to physical phenomena); an example of such a relation is the theory of "bending lines".

   "Stepwise" in this phase addresses the method of mathematical proof: a theorem about relations between knowledge items is established by proof steps, each of which can be justified with respect to some logical calculus. Such proofs are supported by TP, theorem provers. The discipline using TP that way is called "Formal Methods".

2. **Specifying** relates a problem at hand to a body of formalised knowledge: which observations about the problem relate to which kind of knowledge, which items are known in order to solve the problem, what kind of items could be regarded as solutions to the problem, what are their properties ?

   In the example above the problem might be the construction of a balcony, which might be related to the theory of "bending lines", the known items (*input*) might be the function which determines the constant line load q_0 and the *length* of the beam; the output is another function $y(x)$ which determines the bending line, the properties of $y(x)$ are described by the *postcondition*. The result of this phase for the example

[6] STEM abbreviates science, technology, engineering and mathematics as a world-wide concern of educational efforts.

is a so-called "formal specification" as follows:

$$\begin{array}{rcl} input & : & function\ q_0, length\ L \\ precondition & : & q_0\ is\_integrable\_in\ x\ \wedge\ L > 0 \\ output & : & function\ y(x) \\ postcondition & : & y(0) = 0 \wedge y'(0) = 0\ \wedge\ V(0) = q_0.L\ \wedge\ M_b(L) = 0 \end{array}$$

Tool support in this phase concerns searching theories (see for instance [7]), comparing the problem at hand with general patterns of problems (see for instance [8]) and assigning methods (see for instance [9]) solving the specified problem. ISAC can provide user guidance, if (a set of) specifications are prepared for the problem to be exercised.

3. **Solving** starts from a formal specification for the problem at hand, and constructs a solution for the problem following an algorithm determined in the previous phase.

   "Stepwise" solving means that the construction goes on in steps, similar to a calculation with paper and pencil, where each step is justified by some theorem(s) found in the knowledge.

   Tool support in this phase requires a formal specification and a mechanised algorithm solving the problem at hand. And tool support relies on all features and services from above.

These three phases are non-sequential in two directions: First, the phases are iterated during problem solving: observations during "solving" (phase 3) might motivate to revisit "modeling" (phase 1) and revise the theory used, etc. And second the phases come as a cascade, if a problem is divided into sub-problems. For instance, the above example divides the problem of finding bending lines into sub-problems, one of which is integration: when coming to this point in phase 3, the method of integration needs to be selected (phase 2 again), etc. Actually, integration relies on the sub-problem of simplifying fractions — so this example also might apply the dialogues developed in this thesis.

The choice for "fractions" in the case study within the thesis shifts emphasis to one of the phases:

1. *Modeling* is not relevant in the case study, because at the age learners learn to operate on fractions (years 11 to 15), abstract algebra and term rewriting are considered out of scope of mental maturity. However ISAC's knowledge is organised such, that multimedia add-ons is possible to each item (Griesmayer [57]). For instance, the theorem $\frac{a \cdot c}{b \cdot c} = \frac{a}{b} \wedge c \neq 0$ can be accompanied with several kinds of multimedia presentations on cancellation. Such presentations then can be invoked by dialogue guidance.

2. *Specifying* is not relevant in the case study, because learning sequences on fractions focus the domain of rational numbers (and multivariate terms), neglect naturals, integers or complex domains within such sequences and thus avoid explicit specification of the domain for simplification for good reasons. The respective specification is done automatically by ISAC.

3. *Solving* is the only phase relevant in the case study.

The studies' emphasis on phase 3 is in line with current educational practice: Field tests (Neuper [92, 93], Neuper and Reitinger [97]) showed, that teachers appreciated ISAC's

[7] http://www.ist.tugraz.at/projects/isac/www/kbase/thy/index_thy.html
[8] http://www.ist.tugraz.at/projects/isac/www/kbase/pbl/index_pbl.html
[9] http://www.ist.tugraz.at/projects/isac/www/kbase/met/index_met.html

abilities to support stepwise problem *solving*; however, they found it hard to use ISAC's features to let learners explore theories (phase 1) or or let them proceed in trials and errors with variants of algorithms (phase 2), etc. Concerns of the first two phases are preferably taught in ex-cathedra style — so the teachers' (and learners') emphasis is probably a matter of teaching (and learning) style.

2.3 The State of ISAC's Dialogue Component

"One cannot really argue with a mathematical theorem".
Stephen Hawking

As mentioned above, ISAC is designed such that dialogue authoring is not distracted by technicalities of the mathematics-engine or details of the mathematics knowledge. The former technicalities have already sufficiently be covered by §2.2 and §2.3, the latter comprise algebraic simplification (Goldgruber [52]) and cancellation of multivariate polynomials (Karnel [63]).

So only ISAC's dialogue component needs to be considered in more detail. ISAC does not follow the predominant "model view controller" architecture, but separates the dialogue component from the rest of the system (Krempler [73]). The consideration's scope is further narrowed by separating dialogue authoring from Java programming (where ISAC comprises 40.000 lines of Java code; so programming would occupy most resources of a thesis and neglect the specific challenges of dialogue design). This narrowing follows from work by (Kienleitner [64] and (Kober [68]), who integrated a rule- engine into ISAC's dialogue component.

The goals of this work (Kienleitner and Kober [65, 65]) has been summarised as follows:

1. "How can the wealth of research and experience in didactics of mathematics concerning misconceptions made be fruitful for dialogue design ? error-patterns ?"

2. "Designing dialogues relies on trial and error. As long as there are no experiences and no models to learn from, dialogue authors will proceed mainly by trial and error."

3. "[...] establish a data base (DB) structure for retrieving individual performance [...] create a history for analysis by cognitive scientists". This DB structure can be easily used for further user modeling, or understanding better the learners behaviour in ISAC.

The development work in this case study is the first check to which extent the above goals are realised and ready for practice in dialogue authoring. The rule-engine integrated into ISAC's dialogue component is part of a professional expert system (Amador [3]); the engine is free-ware with professional support, but not open source (while all the other components of ISAC are open source with liberal licenses).

Below the detailed consideration of ISAC's dialogue component starts from the users' (and dialogue authors') point of view (while the technicalities will be addressed in §4).

Basic services of the dialogue component are atoms prepared for combination within adaptive user guidance. Prior to the begin of this thesis there was no combination at all, the thesis provided the first rules for how to combine some of the atoms. Below there is a complete list of atoms already implemented in ISAC plus one point, Pt.7, which has been developed due to the suggestions from the side of this thesis. Pt.7 is marked by *italic font*, and so are the points immediately enabled by the implementation of Pt.7, error-patterns.

The list of atoms below takes into account, that the case study only concerns the phase of "solving" as discussed above. This focus allows to omit all the atoms concerned with

the other two phases, and it allows to simplify the presentation (for instance, the notion of "tactics" can be circumvented, because in this phase all tactics concern "rewriting". Thus the notion of "(rewrite-)rule" is sufficient, which is quite intuitive).

1. **the system proposes a formula as a next step** ...
   (a) ... resulting from application of ...
      i. ... one rule (the <next> button)
      ii. ... all rules until ...
         A. ... the final solution is found (the <auto> button)
         B. ... the solution of the current sub-problem is found
         C. ... the next sub-problem begins
   (b) ... *with fill-in gaps determined by a fill-form*
2. **user inputs a formula as a next step**,
   (a) a complete formula and
   the systems gives feed-back: formula is derivable from the context or not
   (b) *a part in a formula with fill-in gaps* and
   the systems gives feed-back: formula is derivable from the current rule
3. **the system proposes rules for application to the current formula**,
   (a) one complete rule
   (b) a list for selection of one rule, where the list
      i. contains applicable rules only
      ii. contains a mix of applicable and not applicable rules
   (c) *a rule with fill-in gaps determined by a fill-form*
4. **the user inputs a rule for application to the current formula**,
   (a) a complete rule
   (b) a rule selected from a list, where the list
      i. contains applicable rules only
      ii. contains a mix of applicable and not applicable rules
   (c) *some or all parts of a rule with fill-in gaps*
5. combinations from Pt.1 and Pt.3 for more detailed help, for instance combination of Pt.3a and Pt.1b in order to help with application of a certain rule, $a + \frac{b}{c} = \frac{a \cdot c}{c} + \frac{b}{c}$, where ... indicate the fill-in gaps in the formula:

$$\begin{array}{l} = \pi + 1 + \frac{1}{3} = \\ \\ = \pi + \frac{1 \cdot \ldots}{\ldots} + \frac{2}{3} \end{array} \qquad a + \frac{b}{c} = \frac{a \cdot c}{c} + \frac{b}{c}$$

6. switch the focus within the lines of a calculation, i.e. determine an arbitrary formula on the worksheet to be the current formula by placing the cursor respectively.
7. *announce the detection of an error-pattern*.

A first survey on the above points shows that the Pt.1 and Pt.3 are symmetric to Pt.2 and Pt.4: the former see the system active (in proposing a formula or a rule), the latter see the learner active (by input of a formula or a rule). This symmetry might indicate a kind of "dialogue between partners on an equal base".

In case the system is active, such activity is triggered by request of the learner in general; exceptions are the first formula in a calculation, which is automatically extracted from the specification by the system. In case of such a requested activity of the system the learner is free to continue with whatever she or he wants: delete the formula, look-up some knowledge, go back to an earlier formula, etc. In case the learner is active, the system's feed-back is immediately connected to the respective input.

Some years ago (Krempler and Neuper [74]) gave an early preview to the possibilities for combining the dialogue atoms to dialogue sequences. However, this work did not anticipate the error-patterns developed within this thesis.

Chapter 3

Contributions from Cognitive Science

"The scientist is not a person who gives the right answers,
he's one who asks the right questions."
Claude Levi-Strauss

Cognitive science is relatively new in the field of mathematics education. It is a challenge to understand the underlying structure of mathematical thinking ([34]), and identify the elements of human cognition which might be applicable for computer education, and very specifically to the educational software: ISAC. Cognitive science claims, that mathematics is a product of adaptive human activities, grounded and embodied in everyday cognitive mechanism, involving the process of abstraction, and that mathematical ideas are quite stable for about hundred years (Nunez [101]). As already mentioned in the introductory chapter: There is a cognitive gap between algebra and arithmetic: humans, and also animals, have basic innate arithmetic, the so called number sense (Bobis [13] and (Gelman and Gallistel [49]). In this chapter we will focus on the aspect of computer supported learning and cognitive science, and so the following questions arise: What is the object of our investigation, and how does mathematical learning and teaching work in a computer-based learning environment. It is important to think of the cognitive processes and involve them in the design for the dialogue. Thus the dialogue shall adapt to the thinking processes, and not vice versa, the human would have to adapt to the software. To reach this goal we need to identify what is important in the learning and teaching process, and how we can answer such an interdisciplinary questions, involving both computer science and human cognition. However there might be serious limitations in regards of technology of background knowledge, we have taken a closer look at many of these questions shortly in the introductory chapter, the guiding goal of this thesis is as follows:

To design the basics of a user-friendly learning environment and a dialogue in ISAC that aims to be compatible how the human mind actually work

Under user-friendly learning environment we understand also non-stressful learning environment, where learners can enjoy mathematical activities without experiencing usual pressure and fear.

Meaning and participants of "Dialogues": According to Oxford Dictionary: "dialogue can be a discussion between two or more people or groups, especially one directed towards exploration of a particular subject or resolution of a problem".(Dictionaries. [42]) Between which of the following groups do we define the dialogue: learner and tutor, or learner and other learner (Considering social dynamics, motivation), or learner and real world (which comprises mathematics among others?)?. In our case the subject is mathematics, and the

resolution of a problem is a specific mathematical task in the domain of fractions. This thesis relates on a dialogue between learner and ISAC, seeing ISAC as a tutor. Second important aspect would be the dialogue between learner and the subject mathematics, which we hope, that through a careful design might be realised to some extend. ISAC is designed to be a learning environment which also contains the tutor (or the teacher) and other peers of the learner, thus serving as a help for establishing a better dialogue between learner and teacher, learner and their peers, and secondary between learner and the real world. Such learning environment is an extremely complex scenario, investigating all aspects would exceed the scope of this thesis, we still found it important to include these aspects shortly into the guidelines showing the possibilities for dialogue authors.

Cognitive Aspects relate to general aspects of learning mathematics (e.g.: cognitive load, working memory), to dialogues in learning mathematics. However, the process of learning mathematics can be very close to other learning processes, the thesis raises the questions: what are the special cognitive aspect relating only to learning mathematics (in our case to fraction), which knowledge, skills are needed to solve an example, and last but not least which are the key cognitive processes relating to errors in the domain of mathematics. The thesis concentrates on the questions of errors, and how to implement these in the dialogue between learner and ISAC. That means we also have to investigate which cognitive aspects relate in a broader sense to dialogues, as part of the learning process, and especially for examples dialogues in learning mathematics, language used in mathematics (e.g.: the different meanings, and ontologies used in mathematics can influence understanding).

However, answering all questions from above goes beyond the scope of this thesis, but as general guidelines, ISAC's design aims to address the following issues:

1. Providing mathematical knowledge, supporting individual learning and the concept of tutoring
2. Supporting stepwise calculation within a selected problem class and providing feedback involving the learner into interaction. In the process of teaching mathematics with a computer software it is very important, that the software gives feedback, reflects on errors (Collins [30]), and provides "answers" for example in form of hints when needed.
3. Furthermore providing a blended learning environment to engage teachers and broader social environments.

Learning Models To which learning models do the above mentioned dialogues fit? Numerous theories and learning models exists, which differ from person to person or school to school (Rogoff et al. [119]): as curriculum-centred, learner-centred (e.g.: Piaget), and sociocultural models (e.g.: Vyogtskys *Zones of Development*), which require different role from both learner (from passive-active-collaborative) and teacher (transmitting curriculum, creating environment for individual learner, matching both individual and collective curricula to learners need).
At current stage of development there is no strong connection in ISAC to the above mentioned theories. Second reason is that ISAC is designed to be a tool for mathematics education, thus fitting various models, and being flexible for adapting other learning environments through dialogue authors. However to state, that the current design is independent from curriculum would go too far, since the topics of fractions and their usual way of solution are addressed in the curriculum, and also at public schools usually we can find the curriculum-centred view, so ISAC has to be able to address this.

In case we have to choose a theory: the design guidelines are more close to the theory the social-cultural theory and to scaffolding: that means to allow learner as much as they can on their own, and intervene and provide assistance when it is needed (Rogoff et al. [119]). This also requires to understand at which knowledge and skill level the learners are we can build up (this is open at the current design, so we decided to leave a freedom of interaction for learners.). in order to "learners develop new cognitive abilities when a teacher leads them through a task-oriented interaction" (Rogoff et al. [119]).

Understanding knowledge-level of the learners is also an open question (Godino [51]) in mathematical education, because it involves the already mentioned, terms, expressions, abstractions, and structure of entities, but a very important question, since the goal to reach in mathematics education is: that learners understand the subject. The problem of understanding is directly linked with problem of knowledge. We have already seen in the previous chapter, what kind of knowledge does ISAC provide. (Godino [51]) states: "Teaching and learning processes should recognize the dialectic duality between the personal and institutional facets of knowledge and understanding".

3.1 Solving Examples in the Domain of Fractions

"With my full philosophical rucksack
I can only climb slowly up the mountain of mathematics."
Ludwig Wittgenstein

In this section we will introduce and solve some examples in the domain of fractions. It is important to show, what are the typical examples, the learners are "struggling with", and to have the same representation about our domain of investigation(fraction). The appendix will provide more examples.

In order to be able to simplify a fraction one has to be able to do basic operation on algebra as well as::

1. Adding/ subtracting
   $2y + x + y + 4x + x = 3y + 6x$
   $-5a - (-7a) = -5a + 7a = 2a$
2. Multiplying Terms:
   $(3a) * (2b) = 6ab$
   $3x^2 * 7x^3 = 21x^5$
   $(a + 3) * (2a + 5) = 2a^2 + 5a + 6a + 15 = 2a^2 + 11a + 15$
3. Factorizing Terms
   $6x + 6 = 6(x + 1)$
   $48x^5 - 12x^3 = 12x^3(x^2 - 1) = 12x^3(x - 1)(x + 1)$
   $(3a) * (2b) = 6ab$

All these basics are required to multiply, divide, add, subtract fractions, and in the end be able to solve an example like that:

1. $\boxed{\frac{3a-6b}{2a+b} * \frac{4a+b}{9a-18b}} = \frac{3(a-2b)}{2a+b} * \frac{4a+b}{9(a-2b)} = \frac{1}{2a+b} * \frac{4a+b}{3} = \frac{4a+b}{6a+3b}$

   Tip: To solve this task one has to be able to factorize: from: $3a - 6b$, to $3(a - 2b)$, then: $9a - 18b = 9(a - 2b)$, which makes possible to cancel with: $3(a - 2b)$.

2. $\boxed{\frac{(1+\frac{1}{x})}{(x^2-1)}}=\frac{(\frac{x}{x}+\frac{1}{x})}{(x-1)(x+1)}=\frac{\frac{x+1}{x}}{(x-1)(x+1)}=\frac{(x+1)}{x(x-1)(x+1)}=\frac{1}{x(x-1)}$
Tip: $1=\frac{x}{x}$ and $x^2-1=(x+1)(x-1)$, this task involves special numbers, and also the ability to recognize patterns, namely the binomial forms, and also requires the knowledge of how to divide and add fraction.

3. $\boxed{\frac{4a^2-16b^2}{18ab^2}*\frac{6a^2}{8a+16b}}=\frac{4a^2-16b^2}{18ab^2}*\frac{6a^2}{8a+16b}=\frac{4(a+2b)(a-2b)}{18ab^2}*\frac{6a^2}{8(a+2b)}=\frac{4(a-2b)}{18ab^2}*\frac{6a^2}{8}=\frac{2(a-2b)}{9b^2}*\frac{3a}{4}=\frac{(a-2b)}{3b^2}*\frac{a}{2}=\frac{(a-2b)a}{6b^2}$
Tip: $4a^2-16b^2=4(a^2-4b^2)=4(a+2b)(a-2b))$ $8a+16b=8(a+2b)$, also involves pattern recognition, cancellation, and factorisation.

The following more difficult example (which is not necessary part of the curriculum) shows that the domain of fractions connects also to other mathematical domains: as equation solving, number theory..etc. Solving such an example requires also some creativity. The example is introduces as a text example, The example is taken from (Daniel [32] and is designed for children aged 13-14.

The task is: **Given the equation with x and y natural numbers, determine the least (positive) value of y such that the equation holds.**

$$\boxed{\frac{1}{\sqrt{y-\sqrt{y^2-1}}}+\frac{1}{\sqrt{y+\sqrt{y^2-1}}}=x}$$

At first sight that might look terrible. One might have an internal pictures of lot of numbers, and their last sign. To be able to solve it, one needs to reform it, first creating the common denominator for the fractions.

$$\frac{\sqrt{y+\sqrt{y^2-1}}}{\sqrt{y-\sqrt{y^2-1}}\cdot\sqrt{y+\sqrt{y^2+1}}}+\frac{\sqrt{y-\sqrt{y^2-1}}}{\sqrt{y+\sqrt{y^2-1}}\cdot\sqrt{y-\sqrt{y^2-1}}}=x$$

since

$$\sqrt{y+\sqrt{y^2-1}}\cdot\sqrt{y-\sqrt{y^2-1}}=\sqrt{y^2-(y^2-1)}=\sqrt{y^2-y^2+1)}=1$$

$$\frac{\sqrt{y+\sqrt{y^2-1}}}{1}+\frac{\sqrt{y-\sqrt{y^2-1}}}{1}=x$$
$$\sqrt{y+\sqrt{y^2-1}}+\sqrt{y-\sqrt{y^2-1}}=x$$

Now we still cannot solve it, we should raise the whole equation at 2.

$$y+\sqrt{y^2-1}+2\cdot 1+y-\sqrt{y^2-1}=x^2$$
$$2\cdot y+2\cdot 1=x^2 \text{ so } 2\cdot(y-1)=x^2$$

Paying attention that x,y are positive natural number, it is sure that we can divide x^2 with 2, so we can also divide x with 2. That means, that x is for sure an even number, and we can make a change of representation in the form of

$$x=2k$$
$$2\cdot(y+1)=4\cdot k^2$$
$$y=2\cdot k^2-1$$

and we already knew that $x=2k$.

This enables to find out the last number of y are: k=1,2,3,4,...etc..

3.2 Knowledge Needed in Mathematics

"Found, but not proven."
Euclides

In the above section we have mentioned the notion of knowledge several time, and showed some for the domain relevant examples. The question is: what kind of mathematical knowledge should we provide in ISAC. With **HINT** pages there is the possibility to provide knowledge in mathematics, and also there is the possibility in ISAC to look up for theorems. What other knowledge could be useful to solve a mathematical problem correctly? Knowledge, the information gained through education plays a crucial role, since depending on learning models, its definition differs through theories (Rogoff et al. [119]).

We have seen, in order to solve examples in a correct way, one needs to have knowledge about mathematics. There are several kinds of mathematical knowledge (Michene [86]): knowledge of items and relations(generalization and specialization, general strategics..etc.), meta-knowledge, epistemological, and representational knowledge, the last "of knowing how to organize and keep track of what one knows such as through maps and networks of items and relations, and a good part of understanding mathematics contains of building up a knowledge base.
The domain of fractions involves both algebra and arithmetic, and we already knew, that arithmetic and algebra does not exactly require the same background knowledge. (McNeil and Alibali [83]) states that mathematics is a domain in which early competence with "basic skills" (arithmetic) is thought to be necessary for advanced thinking and problem solving, and learners learn it many years before they are introduced to complex algebra problems, and because of that learners are well versed in the perceptual patterns of arithmetic problems. But arithmetic and algebra is not all about mathematics. According to (Michene [86]) a mathematician "possesses much more knowledge than that which concerns the deductive aspects of theorems and proofs he has a sense of what to use and when to use it. He has an intuitive feeling forth subject, how it hangs together, and how it relates to other theories." If mathematical knowledge is so many sided and layered, what kind of special knowledge do we need for algebra?

Individual knowledge Individual differences are very well present in mathematics: According to (McNeil and Alibali [83]) this differences depend on perceptual patterns individuals had in the past problem-solving experience, on their strategies (supported also by (Fazio and Siegler [47])).

Knowledge about real and abstract world Mathematics contains abstract symbols which however are abstract representations distanced from any physical references, but according to (Koedinger et al. [70]) there is grounding to real life in abstract symbols. (Mitchelmore and White [89]) distinguishes between levels on abstraction in mathematics and in the process of learning mathematics: "The term abstraction has different meanings in relation to mathematics and the learning of mathematics, and he distinguished between empirical abstraction, which play a key role in the formation of the fundamental mathematical ideas, and he means mathematical objects needs to be linked to real world objects. (Mitchelmore and White [89]) states that mathematical abstraction is firstly based on empirical concepts, and first "children have to learn about the relationships between the empirical concepts", and first they have to learn about this, which is very close to the embodied concepts. This is also supported by (Koedinger et al. [70]) who "demonstrated performance benefits of grounded representations over abstract representationslearners were better at solving simple real life problems than the analogous equations, and they hypothesize that grounded representations are more effective than abstract representations for simpler problems like those typically encountered early in learning". This process can be also called emergent

modelling (Strom et al. [124]).
(Koedinger et al. [70]) speaks about external abstract representation (written on a paper, and "leave out any direct indication of the physical objects and events they refer to"), and internal mental representation , and grounded representation, "that are more concrete and specific, in the sense that they refer to physical objects and everyday events," and he called real-world problem situations as "story problems", which because they refer to familiar objects and also "Besides being more familiar, grounded representations tend to be more reliable, in the sense that learners are less likely to make errors and more likely to detect and correct them when they are made." However even professional mathematician use empirical concepts as an aid to intuition (Boero et al. [14], Devlin [40], Borovik [16]), later in the learning process these empirical concepts are not explicitly present, and the learners get the confidence to operate only on abstract symbols. We will also introduce in a guided example this two levels of abstraction in ISAC, because of course operating on abstract objects in mathematics on complex problems has its advantages: called "symbolic advantage" by (Koedinger et al. [70]). After a time compared to an analogous real-life problem, abstract representation are more effective for more complex problems: "Abstract representations may be more error prone than grounded. Working with abstract representations can be fast and efficient because their concise form allows for quick reading, manipulating, and writing, because they need less working memory than grounded representation."

But on the other side there is strong evidence that many learner "difficulties in learning mathematics can be traced to the fact that, when they learned about an abstract-apart mathematical object, they made no link to the corresponding abstract-general concept" (Mitchelmore and White [88]). Some ideas depend on other abstract concepts, some seem not to have any counterpart in normal experience. In order to have this abstract concepts you need to have a specific amount of knowledge also implemented in ISAC. Referring to the individual aspects of learning mathematics: everybody needs to establish their own connection of basic objects to real world, in order to successfully manipulate mathematical symbols to learn to operate within an abstract system, which as becoming more and more complex, gets more and more "abstract", having less and less connection to real world. In this system: new mathematical objects are constructed by " the establishment of connections, such as inventing a mathematical generalization, proof, or a new strategy of solving a problem ".

But besides taking into consideration the need for connecting mathematical concepts to real world problems: do we know enough about the process of abstraction in order to support this important process in the computer software? According to (Koedinger et al. [70]), if we would know more of the process of abstraction, than in some learning environment we could know when to present empirical/representations things for example before or after formal abstract exercises. This is supported by to (Boero et al. [14]) "we are still far from a comprehensive theoretical answer to the challenge of mathematical abstraction in mathematics education clear response to this challenge would be of great value to researchers and teachers alike."

Knowledge about rules and processes If one wants to solve mathematical examples:one needs to know which rule to applies, and when, or when is a rule not applicable. For example: $5 + 6 = 11$, but $\frac{5}{4} + \frac{6}{3} isnot \frac{11}{7}$. Besides rules, processes, procedures, and skills are also important: based also on the empirical study of with more than 20.000 samples: (Brown and Burton [19]) concluded that learners are "remarkably competent procedure followers, but that they often follow the wrong procedures", which can be the source of errors.

Knowledge from explaining It is important to involve learners in the learning process as much as possible, and guide them (According to (Strom et al. [124]) learners need also to

participate in argument). (Vincent et al. [132]) supports, that one should let learner think, because self-explanations are a very effective meta-cognitive strategies to use in a cognitive tutor in a classroom environment that supports guided learning by doing. Self-speech: "Is the foundation of all higher cognitive powers"(Ben-Zeev [10]). (Williams and Lombrozo [137]) states that explaining plays a key role in learning and generalization: "When learners provide explanations, they learn more effectively and explaining guides learners to interpret what they are learning in terms of unifying patterns or regularities" (Renkl [114]).

Limits of knowledge and the cognitive load Solving a mathematical example successfully might also stop on limits (Sweller [126]): "The cognitive load imposed on a person using a complex problem solving strategy such as means-ends analysis may be an even more important factor in interfering with learning during problem solving." In easier cases the learners simply forgot the rules, and with working one/two times with ISAC, working on their own, they can refresh it, but the program should also provide a basis of understanding mathematical concepts.

In our case we dont involve in ISAC learners on a verbal level, but give the possibility to look up every necessary information on their own, and with the hits, we dont tell the solutions so easily. Self-explanation is a very successful method but hard to implement, thus we will be focus on, how to make an computer guidance, and a paper like solving method. However, It would be impossible at current stage of ISAC to investigate all above mentioned knowledge levels, and to build all into a computer software.

3.3 Systems Used in Mathematics Education

"Life is good for only two things,
discovering mathematics and teaching mathematics"
Simon Poisson

How do other computer software deal with the above defined challenges? This section will introduce some examples on computer software designed, developed for mathematics education, and be shortly compared with ISAC

Buggy System The first computer-based coach (and maybe to ISAC the most close regarding its error diagnosis capacities) was the Buggy System (Brown and Burton [19]) for diagnosing procedural errors in elementary mathematics. In some twenty years ago, there were much more systems that tried to work with errors in mathematics of learners.
(Brown and Burton [19]) created a "diagnostic modelling system for automatically synthesizing a deep-structure model of a learner's misconceptions or bugs in his basic mathematical skills provides a mechanism for explaining why a learner is making a mistake as opposed to simply identifying the mistake with a procedural network". The system worked on the domain of arithmetic and subtraction, where they developed a "computer-based tutoring/gaming system developed to teach learners and learner teachers how to diagnose bugs strategically as well as how to provide a better understanding of the underlying structure of arithmetic skills".
Regarding mathematical knowledge (Brown and Burton [19]) stated that the correct model of an educational software must contain all of the knowledge that can possibly be misunderstood by the learner or else some learner, they tried to realise in Buggy.
The goals of Buggy System is very different to ISAC system. The system ISAC does not try to explain why learners does a certain error. Another difference is, that Buggy did not deal with algebraic problems because that would have needed much more rules (Brown and Burton [19]), however gave a possible direction in building such system with analysing what skills does one need in algebra: "These include not only the generally recognized rules of algebra, but also such normally implicit skills as the reading of formulas, the parsing

of expressions, and the determination of which rules to apply next. This shows, that such education systems can also be used for investigation research question.

Wiris Wiris is a random learning with Moodle used in mathematics curriculum. With Moodle they created random learning questions.
Wiris is used in calculus, algebra, and geometry using quizzes and creates multiple choice questions, but doesnt have tutoring or guidance (Mora et al. [90]).

Falcon Falcon is a CAS (Computer Algebra System) system used in mathematics curriculum by architect learners. CAS deliver solutions without explaining intermediate steps, and are used usually in higher education (Falcon [46]).

T-Algebra T-algebra system (Prank et al. [106]) is on the educational software side a very good example, providing "interactive learning environment for exercises in four areas of school algebra calculation of the values of numerical expressions; operations with fractions; solving of linear equations, inequalities and linear equation systems; operations with monomials and polynomials." T-algebra is also a stepwise solving system (as ISAC), with more than 50 task types (however in ISAC the type of tasks are not limited). The learners can select operations, sub expression, and the system gives feedback and hints. This is a system with error diagnostics recording solution process and mistakes.

Early Algebra Problem Solving (Koedinger and Maclaren [69]) developed in 2002 the so called Early Algebra Problem Solving (EAPS) that contains comprehension processes, represented as ACT-R production rules and predicts learner error-patterns and frequencies in the domain of equation solving based on the idea, that verbal skills of the learners are better than their comprehend skills on equations.

Automath, Macsyma Automath (Bruijn [21]), and Macsyma (Moses [91]) are other older systems on both education and on mathematics theorem proven site. Automath included automated proof checker, with verifying correctness of mathematical theories. The reason Automath did not became successful, that it was not widely publicized. Macsyma is a computer algebra system developed at MIT as part of Project MAC. Macsyma was the first comprehensive symbolic mathematics system and one of the earliest knowledge based systems.

Cognitive Architectures, Cognitive Tutors On cognitive science sides, ACT-R model was often adopted for cognitive tutors, in the educational software e.g.: as of (Koedinger et al. [70]) from Pittsburg Science of Learning Center. A cognitive tutor is an intelligent tutoring system which develops a cognitive model of a learner as he or she interacts with the program, providing problems and individualized instruction based on this model. Some of the most successful applications, like the Cognitive Tutor for Mathematics, are used in thousands of schools across the United States. Some Cognitive Tutors are scenario and mathematical domain related. However, these systems come from instructions, with trying to guess, the difficulties the learner has with. Compared to this system if ISAC find an error it is not only a guess, because it is a more find-grading, we can consider ISAC as a bottom-up design, and we can look at cognitive tutors as top-down design. Cognitive Tutors are the following:

1. Tutors like (Anderson [4], Melis and Siekmann [85]) implement powerful generation of adaptive user-guidance in several aspects, in particular in their reaction to erroneous input during stepwise calculation. The power of this kind of tutors results from their conception which addresses modelling mental processes — which, however, causes multiple efforts, when dealing with basic errors recurring in advanced

procedures; the respective software structure does *not* reflect the systematic built-up of mathematics.

2. Also, tutors based on the concepts of Computer Algebra like (Beeson [9], Prank et al. [106]) very efficiently create user-guidance in stepwise calculation. However, this kind of systems lack logical foundations to combine several procedures of Computer Algebra. The incapability in combining several procedures carries over to error-handling in more complex mathematics problems, for instance in advanced problems in applied mathematics.

3. Additional to the features of these kinds of tutors, the general technology introduced in §2 is able to model the systematic build-up of mathematics and re-uses elementary procedures for implementing advanced procedures. So, for instance, a program for tutoring integral transformations (Ročnik [118]) re-uses the procedure for simplification of fractions, and thus makes all the machinery dealing with errors on fractions available also within the advanced topic.

3.4 Tutoring

"I am always ready to learn,
but I do not always like being taught."
Sir Winston Churchill

The challenge for cognitive science is to understand the underlying structure of mathematical thinking, and identify also the elements applicable for computer education.

In previous chapter we dealt with the questions like: knowledge needed in mathematics in the domain of fractions. This chapter introduced the importance of error-patterns, and argues, that if a computer could support the process of error detection, and correction it would make the work of a teacher easier, and point out the importance of individual tutoring. According to (Payne and Squibb [105]): "achieving "cognitive diagnosis" of learners error may be an important step towards meaningful individualized tutoring" we focus on identifying error patterns necessary to be detected, and corresponding hints to support learners. However in adaptive user guidance not only detection of errors is important, but also the balanced individual feedback. "Balanced" means to give that amount of information learners need to accomplish a current challenge, and not more — accomplishment motivates to tackle the next step in the course of learning by doing. Thus we will correspond this chapter to tutoring, and whether cognitive architecture is needed in ISAC.

Definition for Cognitive Tutoring

According to (Walker [134]): "A cognitive tutor is a type of intelligent tutor that compares learner actions during problem solving to a model of correct problem solving steps and provides context-sensitive hints, error feedback, and individualized problem selection."

According to (Wenger [136]) Cognitive Tutor a kind of intelligent tutoring system, to provide support for guided learning by doing.

Why is tutoring effective?

The "one to one" human tutoring seems to be much more effective over any other teaching method, even compared to traditional classroom teaching. Tutoring was found even more effective than collaborating, and single learning (Collins [30]) about Cognitive Apprenticeship). The question is why is tutoring so effective, and what are its limits, what

are the aspects from single learning and collaborating which are worth to take into consideration or even implement in ISAC environment to gain benefit?
(Chi et al. [26]) seek why human tutoring is so effective, and states, that learning from observing: from actions or behaviour is a very old method of learning, also the one the job apprenticeship learn which is also called imitative learning and tutoring also involves active, constructive interaction. Also in the case learners dont see the underlying mathematics watching at the teachers explanations helps enorm (That is one reason, that we wont completely eliminate during the "dialogue" the *Auto* button from ISAC). Also Vygothsky believed that the ideal learning relationship was that of an apprentice under the tutelage of an expert, and the guided learn provides individualised support for guided learning by doing.
According to (Chi et al. [26]), there are three main hypothesis why tutoring is so important:

1. Tutoring contains pedagogical benefits as: explaining, scaffolding, giving feedback, motivating (ISAC we will concentrate on explaining, giving feedback). The content of tutoring contains the strategic knowledge,methods,reflection, exploration, sequencing: as increasing complexity, diversity.

2. Tutor does typically control lead and dominate the tutoring conversation.

3. However, different learners need different level of collaboration, and type of feedback (that means for example, good learners need less feedback, poor learners require more help, because conduct do probably more errors), tutors are very adaptive to learners and can choose appropriate moves, and know when to interact (This last, is definitely the challenging part in ISAC).

4. According to (Brown and Burton [19]) "One of the greatest talents of tutors is their ability to synthesize an accurate "picture," or model, of a learner's misconceptions from the meagre evidence inherent in his errors."

Can tutoring fail? Results of tutoring depend also on the teacher. "If a tutor cannot accurately access a tutees misunderstanding from a learners perspective, then he cannot be adaptive from this learner-model perspective." So tutoring is only effective if the tutor is very good (Chi et al. [26]). The same is valid for classroom learning/teaching (Bills [11]).

Unfortunately,because its very small teacher to learner ratio schooling replaces apprenticeship like learning. In the future computer based learning environments provide each learner an apprenticeship-like experience. With tutoring we can also focus on cognitive skills. This is the same as the so called Socratic method, where learners have personal contact, and more attention as it is possible in the current class-education. As computer-based learning environments become more pervasive, there is likely to be continued development of new ways to embody these principles in design. The benefit of using software in education is, that people have individual strategies in solving mathematical problems, and also individual ways of learning, also taking into consideration different learning and thinking types. Since we dont know how does abstraction in mathematics work, in the design of the software guiding is an applicable way (Attard and Northcote [5]).
Other need would be to use the cognitive tutor collaborate (Walker [134]). He states, that it is "likely that learners would show further learning gains if they were able to use the tutors collaboratively, and unfortunately these collaborative tools do not occur often in classrooms". We have also thought for the same reasons, that in the learning scenario, ISAC should be designed to be able to do collaborative work, and satisfy a joint way of learning together, under supervision or learning alone.

Cognitive Tutor and Feedback In the dialogue design of ISAC we would like to focus on error detection and feedback, and we would like to investigate how the feedback in tutoring

could looks like.
According (Mathan et al. [82]), learners need feedback in order to develop cognitive skills, as well as error detection to provide adequate feedback. It is also important to provide ready solutions, but to have the quittance, and an external feedback. "Further we take the point of view that the system must provide assistance in the form of feedback on learners explanations and hints in how to explain."
Feedback has also the role of helping by the diagnosis of global misunderstanding (Mathan et al. [82]), and the importance of shave into the learners both the correct procedure and pointing out the incorrect steps.

Need for a Cognitive Architecture? However, this is not in the scope of this thesis, and currently there is no cognitive architecture implemented into the system, ISAC could benefit from the use of a Cognitive Architecture, and this need further investigation.

3.4.1 Feedback and Hints

"Do not train children to learning by force and harshness,
but direct them to it by what amuses their minds."
Plato

In feedback we should also take the following into consideration:

1. Mathematics is directly connected to the language processing (Devlin [40]). Not only with the written, but also the spoken phrases, visual signs, gestures (Nunez [101]). This is involved with representing, analysing, and generalizing patterns, using tables, graphs, words, and symbolic rules relating and comparing different representation,

2. According to (Nunez [101]) we should demystifying proof definitions, formulas, underlying human ideas, because mathematics is created by human, so there are more ways of proofed, formulas are part of language, and can be part of culture.

3. Algebra contains of patterns, patterns recognition, mirroring reflection, order and symmetry, grouping, iteration, linearity, symmetry, transition (Dreyfus [44]).

4. Abstraction process is very important along multiple representation: generalization, operations. (Borovik [16]). (Zhang [141]) states: comparing, combining representations is also very important, since children are better if they allowed to choose their strategies (Graber et al. [54]).

(Mathan et al. [82]) states: traditional intelligent tutoring systems provided feedback on the expert model which means a set of production rules, and the goal is to have an error free performance. On the other side cognitive modelling allows making errors, and provides a guidance. ISAC lies somewhere in-between with its underlying new technological solution. That means that there is a rule model, we havent had implemented a cognitive model (yet). Still the system is able to detect errors, allows making errors, without provide ready solutions, "forces" the learner to think on his/her own, because according to (Mathan et al. [82]) immediate feedback may prevent the development of important skills and cognitive processes.
Hints are important to help learners to proceed, when they cannot rely on other feedback.

How, and when should a system answer? According (Sleeman et al. [122])"Compare different styles of error-based remediation (Swan [125]) found that conflict approach (pointing out errors made by learner and demonstration their consequences) was more effective than a simple reteaching-some found no different, and it was a study with human instructor" ISAC should be able to point out errors, and provide feedback on them.

(Laborde et al. [76]) states, that technology can offer an opportunity for new ways of giving feedback. ISAC' design is ready to support that human cognitive system is fallible and to acknowledge that trials and errors are natural part of the learning process (Sloman [123]): error-patterns etc. could be introduced to ISAC in a straightforward manner, because general and powerful TP-based services are available in ISAC's mathematics engine.

3.5 Guidelines from Cognitive Aspects for the Development of Dialogue System

"In mathematics you don't understand things.
You just get used to them."
Johann von Neumann

During the stage of the dialogue system development, we have to take into consideration also how people interact with information technology.
General question is whether information technology changes the way of thinking of not, because if yes, it should be taken into consideration, that an interaction with a computer triggers different cognitive processes as would be for example by a simple task solving on a traditional paper. One benefit from computer is the off-loading processes, "leading to embowering higher ode thinking processes" (Barzilai and Zohar [8]). (Lajoie [77]) states: there are four types of cognitive tools: that support cognitive and meta-cognitive processes, tools that share the cognitive load by providing support for lower level cognitive skills, tools that engage learners in cognitive activities that would be out of their reach, tools to generate and test hypothesis in the context of problem solving.

(Barzilai and Zohar [8]) conducted an interview with twenty-four academic researches if information technology changed the human way of thinking or not. Part of the results were, "that information technology amplified existing thinking strategies without their nature, or be able to create connections between previous unconnected pieces of data" -the last would be relevant to the current thesis, because ISAC's goal could be: evaluation of information, validation of models.

From (Lajoie [77]) we can take the following: supporting to decrease cognitive load, which might be high in mathematics, support cognitive and meta-cognitive processes, and engage the learners. Technology also might help to compare existing models, validate ideas, see a global picture. It is also challenging to design ISAC to be at the same time rewarding, and at the same challenging.

With educational software we can support the following:

1. Support classroom teaching as well as individual learning. Some features would be hard to realize in the current software, but in this thesis we can provide a basis to identify: where ISAC can support teacher during a class work, based on a one-to-one tutoring and adopting to the learners need.

2. Language: Support both visual language (Pictures, patterns), written language. (On interesting remark: Mathematicians do not use language to think about mathematics (Hadamard [58]). However, gestures and spoken are not part of the current design: implementing gestures (diagrams, pictures connected to gestures) involved in one mathematical concepts could be a possible way

3. We choose to guide in ISAC, and a teacher can provide a guidance without completely understanding where errors originate from, and how mathematical thinking work. Computer environments are also an ideal tool to sport curricular implementing this line of though (Dreyfus [44]).

In ideal case ISAC should react on an error with the following:

1. **Know** if it is only bad counted, or a process/thinking is bad. Most of the times, learners dont have misconceptions, but learn a rule, and over apply it. People are born with a good number sense, human dont tend to loose it, but generally it is hard to connect numbers with abstract symbols.

2. **Show** patterns, next step, half of the next step, hints, rules, good solution

3. **Encourage to make mistakes** and count the bad example till the end, let make repetitions in basics till the learner does it correctly, tell how often a error happens. These errors are necessary to see sometimes where a rule is valid,and where not: Search for errors, count something till the end, even if there is a error, compare good, and bad solution, and look for the why.

The following table shows a comparison between human teacher and possibilities of ISAC from the above mentioned perspectives.

Human Teacher	Computer
Show a good solution, or give a hint	Hints, Fill-Form, Auto, Next
Show strategy	Show tactics/strategy
Show a rule	Show rule
Show differences/parallelism	Giving Hints
Knew what is the error (miscalculating or error)	Hints, Fill-Form error-pattern
Let the learner guide to find the error	Dialogue Modes, Fill-Form
Show special cases (count with concrete numbers)	Giving Hints (Future)
Compare good/bad solution	Future Implementation
Give new easier/similar example/repeat	Future Implementation
Let the learner count even if it is wrong	Future Implementation
Change representations, Make real world example	Future Implementation
Gestures, react on emotion, frustration..etc.	Future Implementation

Table 3.1: Comparing Teacher with Computer and ISAC

Some of the above described belong to the Chapter: "Proposal for Extending ISAC", because at current stage of development some features are not possible to be implement. Analysing the theoretical background: we established the following broader guidelines for ISAC (This section summarises the previous ones by raising some questions this thesis is particularly interested in. These questions are already exemplified by the selection of a topic for realisation in ISAC):

Guidelines for ISAC:

1. Supporting individual learning with different knowledge level, maybe for some answers, only some keywords.
   . . . for all steps in calculations within a selected problem class.

2. Supporting individual way of thinking with multiple strategies (Question is how)
   . . . by accepting all *correct answers to the steps in this problem class.*

3. Supporting that human cognitive system is fallible (responding at error-patterns defined-maybe also two group of error-patterns: one for the error caused by not knowing/remembering the rules, the second due to thinking on wrong concepts)
   . . . with reaction on errors.

4. Taking into consideration, that mathematics is human made, developed through centuries (maybe putting sometimes a small story how the thing developed)
   ... so we are free to define $\frac{1}{0} = 0$*, for instance. And then the system has to demonstrate the consequences by some illustrative examples.*

5. Trying to support the way how humans learn:

   (a) Patterns, repetitions, analogies, connection to different concepts and aspects.
   ... error-patterns deserve in-depth investigation and generalisation.

   (b) Social aspect of learning: trying to support the own thinking, and not influencing too much from a teachers own representation. And the system as guiding.
   ... learning scenarios could realise these aspects.

Scanning the various aspects and challenges in mathematics learning from the side of cognitive science as mentioned above, and scanning the potential of "next-step-guidance" from the side of Computer Mathematics, we identified the following issue as particularly promising from both sides:

Introduction of integration in a calculus course, for instance, captures all attention for advanced concepts and procedures like limit and infinitesimal quantities; however, integration builds upon arithmetic and algebra like addition of fractions, for instance — so frustration might be caused, if errors from the basic level, e.g. in adding fractions, recur on the current level of integration and inhibit success in learning (frustration for both, teachers and learners, because individual tutoring is limited in classes).

As mentioned above, lessons cannot easily be structured alongside errors, errors occur individually for each learner from a large, but finite set of error-patterns as mentioned above, while a teacher hardly can satisfy the individual requests. So the following goal has been established for the thesis ([35]):

Extend the general machinery of "next-step-guidance"
Generalise automated generation of user-guidance
with concurrent detection of errors
recurring during systematic build-up of mathematics. In all math topics and learning scenarios to avoid frustration by basic errors in learning advanced procedures.

Automated generation of user-guidance is successfully accomplished in several tutoring systems, in particular for tutoring symbolic computation of fractions

3.6 Selection of Examples for the Thesis

"I'm a mathematical optimist:
I deal only with positive integers."
Tendai Chitewere

For concurrent detection of errors in ISAC we have to answer the questions if there are typical errors, patterns of errors, are there any models for these, and what are special errors for algebra, and fractions taking cognitive aspects also into consideration.

Are there typical errors, or patterns of errors? This question is important, because if there are no typical patterns, and all errors are random without common features, using error-patterns would be not the optimal choice for ISAC. In mathematics patterns are extremely important (Vogel [133]), and these connect not only to mathematical domains, but also to everyday experience patterns are increasingly important analysing; One could even go so far as to say that identifying and describing patterns is elementary for mathematics. Practising good interacting with patterns supports not only the active learning of mathematics but also a deeper understanding of the world in general. Patterns can be

explored, identified, extended, reproduced, compared, varied, represented, described and created." The thesis tries to make benefit from this understanding. "There are also patterns that teachers can find in analysing the errors learners make during their calculations (error-patterns) as well as patterns that are inherent to mathematical problems.

It might seem, that mathematical errors happen randomly, but (VanLehn et al. [131]) argues that "The realization that errors that appear "random" are often the surface manifestations of a systematic underlying bug is a major conceptual breakthrough for many learner teachers." and in their error detecting system: "Buggy", and later called "Debuggy" analysed thousands of examples, and constructed a catalogue of systematic errors for place-value subtraction (This would be also a relevant application for *ISAC*). Buggy could detect regular patterns of errors "The diagnostician's job is to discover exactly what is the underlying misconception". They stated: "A childs errors are said to be systematic if there exists a procedure that produces his erroneous answers, which was found in nearly all case of them". And they called these precisely defined "erroneous variations of a procedures as bugs". Furthermore they divided between random and systematic errors, and constructed the repair theory, which said, if a learner had unsuccessfully applied a procedure to a given problem, he will attempt a repair, and they also states that the theory has domain independence, but was not applied to other domains.

Of course the question is if it is possible to make patterns, we could use in the *ISAC* program, what how consistent are these errors by learners, how it is likely to make that error on its other occasion? Because if they happen regularly, and are stable, we can help the learners to solve the problem. According to (Brown and VanLehn [20]) two third of the patterns are identifiably consistent patterns of behaviour, and maybe they have hierarchic relation too: error-patterns are quite stable errors.

Small conclusion: In this thesis the errors are represented in form of a pattern, supported by (Brown and VanLehn [20]) in their subtraction analysis, which states: that "errors are quite clear patterns". The above mentioned example dealt mostly with subtraction, now we will investigate which error-patterns are common in the domain of algebra and fraction.

3.6.1 Models of Errors and Towards of Computerization?

"Real mathematics must be justified as art
if it can be justified at all."
G.H. Hardy

Modelling errors? The causes of errors are not absolutely clear yet, especially from cognitive science point of view, and the question for us is whether we need model for misconceptions in order to teach learners with the computer? According to (Brown and Burton [19]) "a detailed model of a learner's knowledge, including his misconceptions, is a prerequisite to successful re-mediation". Also according to (Balacheff [7]) "All the predefines chapters make it clear that, beyond merely observing learners errors while they performed some mathematical task-that is to say considering errors in facts that we can count and classify,- we need to propose explanations of their origins." Before planning the system, and the errors, at least we have to take a look on their possible origin. For example as we mentioned before the importance of multiple representation, many learners struggle with these feature of fractions (Bills [11]). Usually children do a lot of errors in using "algebraic motion". From educational point of view to understand what is missing, we have to focus more on this algebraic thinking. It is very important is to identify exactly the variations of internal structure of mathematics that generate the misconceptions (Lakoff and Nunez [79]).

According to (Brown and Burton [19] "children make errors because they have faulty versions of certain components and skills. Errors are made because certain component is missing, and we should examine their relationship to the correct one". (We would like to make the readers attention for this sentence, because later in the design we will see it again.) Models can also provide some problems. (Brown and Burton [19]) faced the following problems: It was often difficult to infer a learners bug from his answers, also combinations of primitive bugs would not be detected, or random mistakes. "Buggy produces models that behave functionally as the..., these models are not very convincing as psephological models" (Young and OShea [140].

However, the originating of misconceptions or errors depends on a lot of factor (individual, social, cognitive factors), like how the teacher explains the task (Classrooms activity in the early year focus on mathematical product rather than to mathematical processes! This can be also a reason, that it would be much better for some skills to be developed in the earlier years, especially in the domain of algebra), what kind of "inexistent" rules the learner creates, due to processes which cognitively work, depending also on previous knowledge and other domains, problem solving strategies. Usually introducing realistic examples, the number of mistakes can be decreased
Generally in mathematics according to (Brown and Burton [19], VanLehn [130]) children make errors because the task contains something they have never learned, forgot, or, miss some skills, or, learned something else, or, have representation problems.

It is still an open question, whether we will be able to design a system which can detect, and originate the cause of errors regarding the focus on: what the learner is thinking.

Production System model and Repair theory There are several models of errors existing on the Production system models (Young and OShea [140]), which contains a collection of rules mixtures of different strategies. Other theory is the so called repair theory: "a situation which the core procedure is unable to proceed, and deleting the individual rules" Child might switch between a number of different bugs even within the course by a simple script. (Brown and Burton [19], VanLehn et al. [131]) deleted steps from procedure (Brown and Burton [19]) called "bug migration" with the same type of task, the learner may display different bugs both during the same test-period and between different tests. "learners store the path, and merely uses it with the new task. Error detection can be also seen as the notion of debugging: which means also to identify the but, and to repair the bug.

Diagnostic Model (Brown and Burton [19]) distinguished between procedural knowledge and skills, and intended to recognize both with the software. "We introduce the term diagnostic model that mean a representation of a learner's procedural knowledge or skill that depicts his internalization of a skill as a variant of a correct version of that skill.The breakdown of the skill into shared sub skills can also account for the recurrence of similar errors in different skills." They build a diagnostic model.

Cognitive Tutors, Cognitive Architecture, Cognitive Theory :
Cognitive Tutors (As mentioned in previous chapter) are grounded in the ACT-R theory of cognition and learning. (Payne and Squibb [105]) states: "Central claim is that many errors can be explained by the learners mental representation and application of a faulty procedure, called either a bug or a "mal" rule they can be generated by some plausible cognitive theory."

Leeds Modelling System According to (Sleeman et al. [122])" ITS intelligent tutoring system reteaching us re mediation diagnosis of the errors in more difficult than re mediation. If one inferred an accurate model of learner error it is than relatively straightforward to me that model to direct a re-mediate dialogue reteaching = reteaching the correct methods" (Sleeman [121]) created the Leeds Modelling System (lms) for learners error.. LMS

is a rule based databases (with diagnostic capacities for the majority of errors)and a process oriented explanation for learner error which generative mechanism uses a collection of primitive bugs. "First the practical aim by producing teaching systems which are truly adaptive to the needs of the learners, and second the theoretical interest involved in formulating these activities as algorithm.

We dont want to explain in the first hand why these errors occurs so we have chosen a rather "safe" way: and grouped the set of errors in a way, that in the cases we surely knew which should have been the correct solution. There might be an underlying model, structure, hierarchy as mentioned before, but this fine-grading level we take the possibility to build up later further analysis. Our advantage is that we connect errors/errors to the correct solutions. Before the implementation is described, we consider design issues, because there is for instance an abundance of learners' errors concerning fractions described in the literature. Many of them have been experienced by the authors as well.

3.7 Error-Patterns in Algebra and in the Domain of Fractions?

"Anyone who has never made a mistake has never tried anything new."
Albert Einstein

Learning mathematics is not always easy. At which time one has to apply a specific rule is crucial in solving an example successfully.
It would be important to know, and also would be a great benefit of ISAC, if we could locate, and individually detect at which part (e.g.: Due to rules, do to patterns..etc.) do learner have problems. In order to investigate the question lets see in more details, where are the typical errors, and what happens there, because this will influence our design decisions on choosing error-patterns.
(Koedinger et al. [70]) state "learners errors in the symbolic format often revealed serious difficulties with the syntax and semantics." or learners made errors in comprehending and manipulating algebraic expressions (Sleeman [121]).

To be successful in algebra you have to be able to operate for example with different objects, you have to know what are numbers, variables, and what are their features:

Objects

1. **Numbers**(e.g.: $1, 3, 3/4, 0, 4..etc$), Can be added with numbers, can be changed in forms. (e.g.: fraction, decimals..etc.)

2. **Variables** (e.g.: $a, b.x, x^{3,ax,..etc.}$) Variable are more complex, because they represent a concrete number, or even set of numbers like, $x = a = b$, They can represent the same set of numbers/number, or not. Variable have names. It is possible to choose their names as one likes, but not always (e.g.: like in an equation, if I change the name of the x, I have to change all name of the x.) (Not mentioned parameters which are somewhere in-between a number and a variable).
   Also hard is to learn, which variable are the same, like x cannot be added to x^2, but they represent the same possible number(s). This is a contradiction, the brain has to deal with, and one has to learn.
   Numbers and variables appear usually together in terms: $2 \cdot x + 3 \cdot x = 5 \cdot x$
   Errors in variables, numbers belong also to identified learners difficulties with algebraic symbols (Clement [28], McNeil and Alibali [83]):

   (a) For example: $3m + 6$ could be interpreted as $9m$ or,

(b) $a+b+ab+a+b$ could also make problem, and could be simplified as $3a+3b$, because

(c) the meaning of letters, and numbers might be confusing.

Operations: +, -, /, *, () What are operators is not so easy to understand. Sometimes the equal sign could be also interpreted as a logical connector (Clement [28], McNeil and Alibali [83]).

1. $3(m+2)$ could be also 9 or
2. $\frac{1}{2}+\frac{1}{4}=\frac{2}{6}$
3. $\frac{1}{a}+\frac{1}{b}=\frac{1}{a+b}$
4. $a*a*a=3a$ however $3a=a+a+a$

Patterns: (e.g.: $(a+b)^2=a^2+2ab+b^2$)

Rules (e.g.: Commutativity: $a+b=b+a$, $a*b=b*a$ Associativity $(ab)c=a(bc)$, $(a+b)+c=a+(b+c)$, Distributivity: $a*(b+c)=ab+ac$,)
Operation have basic priority rules: Which operation are to be carried out first should be grounded in an early educational level:

1. $3+(2/5)$ is not the same as $(3+2)/5$
2. Sometimes things disappear (Special cases)
3. $1*a=1a$, but $2*3$ is not 23
4. $0*a=0=$ .....
5. Sometimes different variables are the same
   $a=b=c=d$
6. Sometimes same variables are different
   you cannot add these: $\frac{x^2}{5}+\frac{x^3}{5}$

End-result: For some learner it is also confusing, what is an end result, when to stop solving an example(Tirosh et al. [128]). The already mentioned $3m+6$ just simply could not look like an end result, and could be simplified further.

Where do errors of algebra and fractions originate? Error-patterns of fractions related of course to the error-patterns of algebra. If we speak about errors, what kind of errors can happen during learning fraction, when do typical errors happen? These errors of fractions are often systematic and rule-based rather than random: (Yetkin [139], Giaquinto [50]) average learners around the world have difficulties in learning about fraction. One of the problem according to (Giaquinto [50]) is, that it is very difficult in fractions that many properties are not any more true for all numbers."For example, with fractions, multiplication does not always lead to an answer larger than the multiplicands, division does not always lead to an answer smaller than the dividend, and numbers do not have unique successors. Many typical errors happen with dealing with the intuitive special numbers, 1 and 0. For some it can be very confusing if the fractions are bigger than the whole, or change from or to mixed form. It can be also very hard to compare numbers, change between forms, because the same denominator is not the easier concept, also understanding which fractions are the same is crucial in a successful solving of example.

We have chosen the following domains of errors (and causes) to be relevant for this thesis:

1. **Problems with representation**:
   error-patterns may originate due to thinking on different representation (e.g.: proportion, division, fraction). (Fazio and Siegler [47]) states, that according to age you need different representations and support different strategies in teaching fraction. For example using the results that children understand the concept of equal sharing, proportional relations.

2. **Problems with different levels of abstraction**:
   Learner may correctly answer a problem, and give a wrong solution. These problems might arise because of the connections between real-world problems and fraction notation. Transforming text into mathematical symbols is very challenging task.

3. **Numerator and Denominator are not the same**:
   Usual problem is: Treating fractions numerators and denominators as separate whole numbers instead of treating fractions as unified number. Thus leading to one of the most common errors: Leaving the denominator unchanged. (Fazio and Siegler [47]), as well as to errors in addition, subtraction of fraction.

4. **Misunderstanding mixed numbers**:
   "Some learners ignore the fractional parts and focus only on the whole number. Others decide that the whole numbers have the same denominator as the fractions, or adding the whole number to the numerator of the fractional part"(Fazio and Siegler [47]).

5. **Misunderstanding exact values of fraction**:

   (a) What is exactly the value of the fractions: for example "being confused of the meaning of fraction larger than 1"

   (b) Comparing fractions

   (c) What are the equal numbers?: $\frac{1}{2}$ of 6, or 0.5 of 6, or 50 percent of 6 are all the same. decimals and fractions are different types of numbers. Changing between forms might be challenging, and also not obvious for the first sight,or when it is really needed.

6. **Problems due to learning process**:
   According to (Ma [81]) most learners modify their errors as time goes by even though they make errors about already learned contents. "Children often confuse the rules of whole number arithmetic with those of fraction arithmetic. Learners will gain greater conceptual understanding of fraction arithmetic when they understand why the procedures from whole numbers do not work, rather than just learning a new procedure for fractions (Ma [81]).

7. **Problems with Operators**:
   According to (Davis et al. [36]) algebra learners tend to overgeneralise from instances using "old" operator instead of a more recently introduces one. Operators involve multiplication, division, addition, subtraction:

   (a) **Multiplication**:
   Usually multiplication increases a number, but in fractions it can happen both, because multiplying fractions may mean finding a "fraction of a fraction" so multiplication can both increase or decrease a number.

(b) **Division**:
Dividing a whole numbers by fractions means e.g.: $3 : \frac{1}{4} = \frac{3}{4}$.
(Ma [81]) analysed empirically the repeated error-patterns in the division of fractions by elementary learners in Seoul. Their conclusion was: "most learners that continually make errors cause reciprocals of natural numbers in the divisor when calculating on (fraction) = (natural number) secondly most learners recognize that the divisor has to change the reciprocal." The meaning or representation of division is more complex: dividing by fractions mean how many times the divisor can go into the dividend. Additionally: "Few learners understand why one inverts a fractional divisor before multiplying" (Carraher [24]).

(c) **Addition and subtraction:**
By adding fractions it is very important to understand what is a common denominator: "Children who understand why a common denominator is necessary when adding fractions are more likely to remember the correct procedure than children who do not understand why common denominators are required."The common error of trying to add fractions by first adding the numerators and then adding the denominators stems in part from not understanding that fractions are numbers with magnitudes." This is a typical example, that a previous learned operations (in that case addition $1 + 3 = 4$) dominates over the new learned rule (common denominator).
Also the error-patterns on the addition and subtraction of fractions were studied empirically in Malaysia (Idris and Narayanan [61]). "The findings indicate that errors in addition operations are 29.8 percent of careless errors, negligence errors 26.3 percent and 11.1 percent of systematic random errors. In systematic errors, 50.6 percent of learners have a problem converting to the lowest common denominator, 26.2 percent encounter problems in the process of understanding, and 14.9 percent have problems dealing with improper fractions. As for the subtraction of fractions, there are 26.4 percent or systematic errors, 10.3 percent of careless errors, and 2.5 percent of random errors. In systematic errors, 47.9 percent of of learners faced problems in the process of understanding." The reason might be, that automatically what you see, so the visual input is stronger, than understanding what exactly 1/3 and 1/4 means.

8. **Special cases:**

(a) 0, 1: In the work of (Brown and VanLehn [20]) the majority of the bugs/errors had to do with zero. (If this is valid also to other domains, than we could hypothesis, that also by fractions one of the main errors would be due to special numbers as 0, and 1).

(b) negative numbers (e.g.: $-8 + 2 = 6$)

(c) binomial formats are very often present in fraction examples.

Short conclusion: Why do these errors in fractions happen? First of all, this is an open question (Breiteig and Grevholm [18]). Further cognitive research should be needed to find out on how learners develop their understanding of fraction: modelling and developing, what is a whole, partitioning, comparing, number lines, density, equivalent fraction, addition and substitution, directly implementing causes would be an early step to ISAC, but we can include this causes of errors in the design of the feedback.

3.8 Error-Patterns for $\mathcal{ISAC}$

"Experience is simply the name we give our mistakes."
Oscar Wilde

From previous chapter we can conclude, that it is possible to create error-patterns for detecting errors in $\mathcal{ISAC}$, because most of the errors are clearly pattern like. The causes of such errors, and errors in the domain of fractions compiled a long list, which then can be separated into groups based on the causes of such error. Separation into groups is important because several different patterns can be related to the same cause of such errors (e.g.: there are more clear error-patterns in addition of fraction), Error-patterns, their belonging to a group in $\mathcal{ISAC}$ is in principle a delicate design decision by the dialogue author. In our case we made up decisions about: the kind and amount of error-patterns, and their separation. By separating error-pattern we mean grouping them. Groups make it possible to give at some point of the interaction the same feedback (e.g.: hint-pages from Table 4.1).

The clear advantage of error-patterns is, that we can define exact points in the calculation where we knew for sure what went wrong. In our case they also mean, what should have been the good solution (e.g.: in addition, see Table 4.1 the learner should have carried out an addition but for some reason he/she entered an in-correct formula), and we can give a feedback and in a form of a hint-page explain how a successful addition looks like.In some cases it might be hard to decide which error-pattern to which groups belong (see group no. **4**: `Special cases with 0` and group no. **5**: `Special cases with 1 or -1`). In our case we collected error-patterns from typical errors by learners(e.g.: these may happen during examples involving special numbers (0,1), or in the transition between different domains as addition can be confused with multiplication or vice versa). Advantage is, that in this way we can grasp points, where we surely can give a hint for the learner.

Having too much error-patterns is costly in resources, because it would slow down the calculation process, so we have to choose carefully which and how much error-patterns we choose.

In our case we made up the following groups of error-patterns (Each group shows some example from the group, and is incomplete, see further details in the Appendix):

1. **Addition or Subtraction** $\frac{a}{c} \pm \frac{b}{c} = \frac{a \pm b}{2 \cdot c}$, $\frac{a}{b} \pm \frac{c}{d} = \frac{a \pm c}{b \pm d}$, $\frac{a}{b} \pm \frac{c}{d} = \frac{a \pm c}{b * d}$, $\frac{a}{b} \pm \frac{c}{d} = \frac{a * c}{b * d}$, $\frac{a}{b} \pm \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$, $a \pm \frac{b}{c} = \frac{a \pm b}{c}$, $a \pm \frac{b}{c} = \frac{a \cdot b}{c}$ These errors might result from not remembering, not knowing (..etc) that different operational rules are valid for numerator and denominator. Subtraction is very similar to addition.

2. **Multiplication**: $\frac{a}{c} \cdot \frac{b}{c} = \frac{a \cdot b}{c}$, $\frac{c}{a - b} \cdot \frac{d}{c} = \frac{c \cdot d}{a - b \cdot c}$, $\frac{a}{b} \cdot \frac{c}{b} = \frac{a + c}{b}$ Because in the domain of fractions the learner might confuse multiplication with addition we found it important to separate them into two groups.

3. **Division**: $\frac{a}{b} \div \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$, $\frac{a}{b} \div \frac{c}{d} = \frac{a \div c}{b \cdot d}$ We grouped divisions in a different group, because the rule of solving these examples differs significantly from addition and multiplication, and involves the mathematical notion of reciprocal.

4. **Cancellation of Fraction**: $\frac{a+b}{a+c} = \frac{b}{c}$, $\frac{a+b}{a} = b$, $\frac{a \cdot c + b}{a} = b + c$, $\frac{a + bx}{c + dx} = \frac{a+b}{c+d}$ In order to successfully simplify rational terms, the learner should be able first to factorize, and also understand that cancelling off only a portion of a factor would not lead to a correct result.

5. **Factorization**: $c + c \cdot b = c \cdot (0 + b)$ These rules not necessarily connect strongly to fraction, but also to other domains. Still many learners struggle with them, so we found necessary to mention them in this domain.

6. **Special cases with 0**: $\frac{b}{c} - \frac{a}{c} = \frac{b-a}{0}$, $\frac{a}{a} = 0$ Zero is a challenge because it results from embodied ideas (Nunez [101], represents existing nothing, both involves many special rules a learner should know (e.g.: Zero, divided by any non-zero number is zero, but the division with zero as the denominator is not defined...etc.)

7. **Special cases with 1 or (-1)**: $\frac{a-b}{b-a} = 1$, $-\frac{a}{-a} = -1$ The Number 1 results also from embodied experiences, represents the whole, some of the special rules to remember are for example: Any non-zero number divided by itself equals one, Multiplying a number by one does not change its value..etc.

8. **Binomial forms**: $\frac{a-b}{a^2-b^2} = a - b$, $(a-b)^2 = a^2 - b^2$, $(a+b)^2 = a^2 + b^2$ We took binomial forms in the error-patterns, because usually fractions involving variables always contain one of the tree listed binomial terms. In this case not remembering the adequate rule might cause problems.

9. **Changing between forms**: $a\frac{b}{c} = \frac{a+b}{c}$, $a\frac{b}{c} = \frac{a \cdot d+b}{c}$ Changing between the different forms (mixed form, decimal number, simple fraction) is also challenging.

10. **Priority Rules**: $a - b \cdot c = (a - b) \cdot c$ Similar to factorization this is also a rather basic rule, and we will not focus on them in this thesis.

These groups are result of several design decisions. For instance, both groups: addition, and multiplication, address multiplication *and* addition: addition can be confused with multiplication or vice versa, so a dialogue author has to decide where a certain error fits better.

Chapter 4

Implementation in ISAC Modelling "Next Step Guidance"

"If people do not believe that mathematics is simple,
it is only because they do not realize how complicated life is."
John von Neumann

The choice for the mathematical domain to concretely work on within this thesis was free, in principle. In fact the choice was limited to the domains already implemented in the ISAC prototype. Solving problems with fractions involves complex cognitive abilities (Kieran [66], one can vary the form of them, e.g. from a mixed form $2\frac{3}{5}$ to $\frac{13}{5}$ or 2.6, one can carry out different operations (addition, multiplication, etc.). Also, fractions establish relationship between two numbers, involve multiple representation (proportion, linear change, simple value, etc). This chapter will describe the kinds of dialogues to be modelled in ISAC, the definitions required by ISAC's mathematics engine to automatically detect error-patterns and to create specific formulas (fill-forms) and hint-pages for help, and the rules telling ISAC's dialogue module how to react in case of error-patterns.

4.1 Example Dialogues for Fractions

"Mathematicians do not study objects,
but relations between objects."
Henri Poincare

It is out of scope of this thesis to show how learners develop mental representations of fractions, rather the goal is to develop an interactive dialogue between a learner and software (Ritter et al. [115]) — and this goal also conforms to the quote of Henri Poincare above.

For a better overview we would like to represent the dialogue in the Fig.4.1 about the interaction planned and implemented in ISAC. In Fig.4.1 we see an example where a learner enters an error, which is detected by ISAC providing a hint in the first step. In case the learner has no idea what to do next, he/she can request help, and as a response, gets the fill-forms, where learners have the only possibility to fill-in a term contain space-holders. These are connected to error-patterns, and activated at clicking on a *HELP* button. The reason for introducing fill-forms is: instead of providing the learner the whole correct next step, we would like to engage the learner to try solving the example on their own (according to theoretical background described in previous chapters), and ISAC shows only the part of the solutions, and the missing parts (space holders) are to be filled out by the learner. In case an error-pattern is detected: first a *HINT* page pups up, and there is no possibility to

request the whole next step by hitting *NEXT* button, or an automatic solution by the *AUTO* button. If the learner still cannot solve the example, he or she may still request help by clicking on the newly appeared *HELP* button, which activates the fill-form. (During the whole process it is still possible to browse in the knowledge materials of ISAC) Depending on how many fill-forms in ISAC for this error-pattern exists, the learner can request further help, and only as a last step the whole correct next step appears. This process is shown in Fig. 4.1.

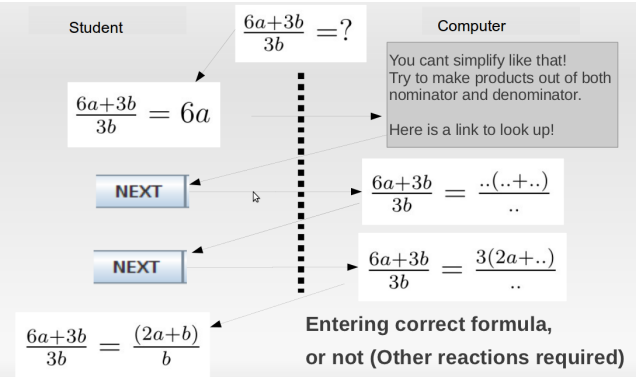

Figure 4.1: Interaction with ISAC

It might also happen, that during this process an other Error-Pattern is detected. A more detailed example can be seen in the Fig. 4.2, which represents a screen shot from ISAC.

For mechanised dialogues we design how to react on what the learner is doing. For that purpose we need to collect information about the learners action. In ISAC this is information the learner is looking up, the formula he/she is typing in during calculation, etc. An input formula might be correct or incorrect. In the case of a correct formula, there is not much to do, in the case of an incorrect formula we can give feedback. The interaction is based on reflection, and the dialogue gives hints if the learner requests for. In Fig.4.2, as a feedback for the (incorrect) input $\frac{8 \cdot x}{8 \cdot y}$, a "hint-page" popped up with the hint `"This is not the correct way of addition. Please look up here !"`. Since ISAC uses HTML technology, dialogue authors will be free to exploit the ever increasing features of HTML. In the right upper corner Fig.4.2 indicates buttons changing during the subsequent interactions. The case of input of an incorrect formula shown in the screen-shot on Fig.4.2.

This example will proceed from the situation in Fig.4.2 to interactions, where the learner requests more help and the software presents "fill-forms" with place-holders to be completed by the learner: $\frac{..+..}{..}$, $\frac{..+..}{4y}$, or $\frac{5x+..}{4y}$.

Tab.4.1 shows the situation in Fig.4.2 in lines `00..02`. Line `03` assumes the learner pushing the HELP button appearing instead of NEXT and AUTO (which would allow to request the next step or even the final result from the system). Lines `04..05` show the fill-form requested by the learner, but not filled correctly. So in line `06` the system presents the tactics applying $\frac{a}{c} + \frac{b}{c} = \frac{a+b}{c}$ which leads to a correct input by the learner finally. `rule 1-4` in the right-most column will be discussed §4.7.

In our opinion the examples above show that it is possible to come closer to a paper-like learning experience with more interaction.

In the next sections we explain, how the technological background works: especially error-patterns, hint-pages, Fill-Forms, Fill-Patterns, and Rules, and how these establish the prerequisites for dialogue guidance.

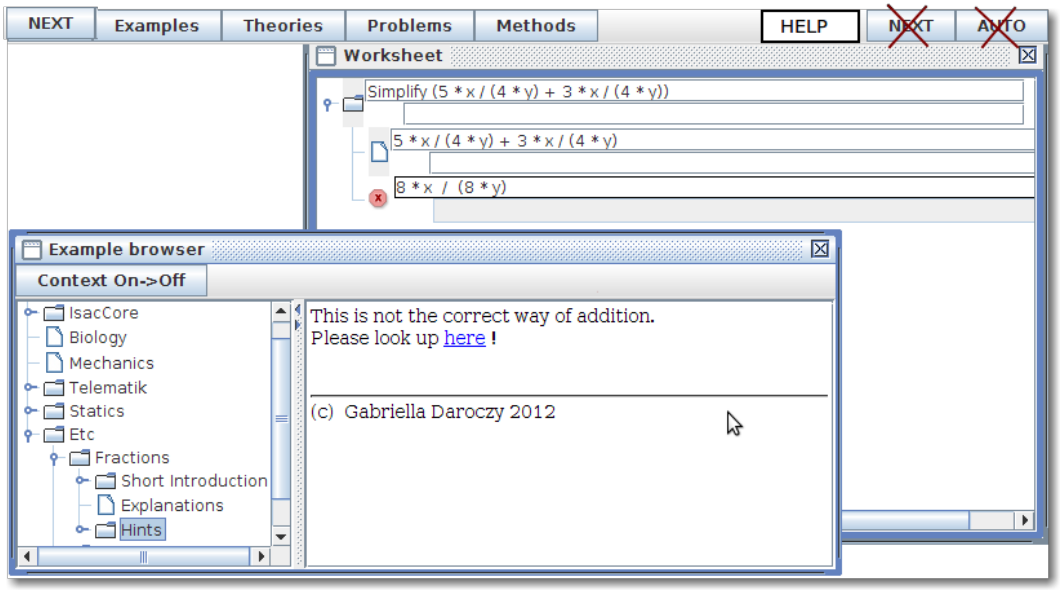

Figure 4.2: ISAC's Front-End with Incorrect Input

no.	learner's action	user-interface	system's action	dialogue-rule
00		$\frac{5x}{4y} + \frac{3x}{4y} =$		
01	types in $\longrightarrow$	$\frac{8x}{8y}$		
02		hint-page 1	$\longleftarrow$ pops up	`rule 1`
03	hit help button $\longrightarrow$	$\frac{..+..}{} =$	$\longleftarrow$ show fill-form	`rule 2`
04	hit help button $\longrightarrow$	$\frac{..+..}{4y} =$	$\longleftarrow$ next fill-form	`rule 3`
05	hit help button $\longrightarrow$	$\frac{5x+..}{4y} =$	$\longleftarrow$ next fill-form	`rule 3`
06	hit help button $\longrightarrow$	Tactics	$\longleftarrow$ no more fill-form	`rule 4`
07	types in $\longrightarrow$	$\frac{8x}{4y}$	correct	
...				

Table 4.1: Interactions for EP "add-fractions"

4.2 Error-Patterns

"How is an error possible in mathematics?"
Henri Poincare

This section describes how the prototype of ISAC detects the error-patterns above, how the counters are connected to them (and to fill-patterns); examples of error-patterns are described exactly as implemented in ISAC's mathematics engine.

4.2.1 Information, Designing Counters

"The essence of mathematics is not to make simple things complicated, but to make complicated things simple."
S. Gudder

After examining the first question of the thesis: if there are typical even systematic (that means non-random) error-patterns we can use, and detect, the second question arises: what do we do with these error patters? Sure, we have to react on it, but could we get an extra

benefit of using statistical analysis of the error-patterns, or some additional benefit from the so called "counters"? In this chapter we will examine the question, how counters could be taken into account in designing the dialogue more efficiently, and what would be their the benefit. The overall goal is to keep the interaction, and the learning experiences a good experience, and an effective process for the learner, and also to have tools in the hands of the dialogue authors, which enables them to design the learning scenario as they require, and to design ISAC react in a requested way, and time.

In designing dialogues it is also important what can be observed (e.g.: which information can we collect), described, and how, and what is prescribed by the software before and after implementing error-patterns. During the interaction with ISAC we can observe for example what the learner is clicking on, if he/she requests help, e.g.: how often does conduct a specific error-pattern. On one hand we have the possibility to collect as many information as possible from the learner to design further interaction possibilities. Counters could exactly serve this purpose. Before introducing counters in details, we have to take into account several questions. For example: How do we know for example if the learner only miscalculated something, or needs a detailed explanation? It is easy to see, that if ISAC provides a long detailed explanation in the first case, this would be as demotivating as making a small hint in the second case. One way, which needs validation to find out what is a miscalculation only: is maybe the frequency of error-patterns. Miscalculation might happen fewer times, than real errors. This chapter will introduce the counters, but we should take into consideration, that this design of counters is only one possibility.

Design addresses part of mathematics as well as part of learning mathematics in respect of knowledge needed, whether error-pattern exists and can be used for ISAC, and cognitive science and dialogues in respect of processes on individual and social level involved tutoring and what kind of feedback, hints are needed, comparing also other computer education systems, and human interaction to address also design guidelines.

For the counters in ISAC we will take into account the following guiding design questions:

Questions about information Where can we collect information, what is ISAC automatically collecting, which can be used. As in details:
What kind of information is already stored in the system, and what do we want to store during a session?
Answer: In ISAC there is a database, which stores all possible interaction, e.g.: clicks. So the questions is whether to make (and if yes, which information) this accessible during the interaction/session for dialogue designers or not.
During a current session we want to store as many information as possible because we are interested in the whole learning process, and we are interested about any kind of information the learner is doing, in order to make "judgement" later: like did the learner learn the current task? How well did he/she perform, where are the points, where there is a need for a different interaction,more help, or what should he/she repeat?
However we cant focus on all of the information, because that would be contra-productive, and the question is which information will we be focusing on?

What are time-intervals we would like to focus on? Answer: Basically there are three big time-interval when we can collect information, the first is during a concrete task (an example is open in a window), during a session (this contains more examples which can be opened parallel, and also after one another), or during the entire learning process (this is dedicated to one learner, and would make the possibility for example to come back to examples, which were not solved successfully in the last session, and the next session could start with capturing in a different learning mode).

1. **During concrete task:**
   Concrete task means if the user opens an example in a windows in order to solve it, e.g.: simplify$(6a+3b)/3a$. It is possible, that the user may have opened several tasks parallel, so it is important, he/she can switch between each of these task. It matters in what kind of an example does he/she conduct an error or requests a help.

2. **During a session:**
   Session is a period between logging in and logging out. During one session the learner may open several tasks, solve them, or close, look up for help, request hints..etc. The current thesis focuses on the counters which we can get from a single session. We have to make distinction in ISAC in which task.

3. **During the whole learning scenario:**
   The whole interaction contains all learning session, made up from several sessions and belong to a single user. This is important, because in this way we can make later a user profile, and record what the learner may improve, or forget things during sessions. However planning a whole learning scenario is not the core question of this thesis, but maybe in the close future it might be interesting to check if a learner remembers learned examples, or can solve a similar example struggled with in the previous session, or get back to problems he/she could not solve yet,because learning might also happen during a resting period.

What else do we have to take into consideration? For example: how do we know, that an example is successfully or not successfully solved?
With the following list we try to categorizes some possibilities ISAC could take into account, and their numbers could be important: **Number of opened examples**
These includes all examples the learner opened, including all examples where he/she entered a formula, and including all examples where not. Some cases might also happen where he/she is searching for an example she likes to solve, opens more of them, and chooses in the end only one or some of them.

1. **Number of begin examples**
   Examples where the learner entered a formula:the number where he/she tried to enter a solution

2. **Successfully solved tasks**
   Examples solved alone, or tasks solved with the auto button. The question is if we should add the number of examples solved with the auto button to the number of successfully solved task. It might be the case that the learner did see the solution and also might understand it. It would be an other possibility to give back the example solved with the auto button for a later check. It is also a design question what we call as: the tasks solved alone.

3. **Unsolved task**
   Unsolved task are also design question, because what do we call as unsolved example: if the last entering is wrong, and the example was closed, or the learner logged out of the session?
   Under such design questions, would it make sense at all to count for example the ratio of successfully solved and unsolved tasks? In the next point we will list up the interactions where we have very less information, and we can have only a guess:

What can we count at all during a session, and what can we not count? The first, and the easiest and in this stage of ISAC development the most useful information we get during an interaction, is if the learner enters a correct or a non correct formula, but we have no information about:

1. If the learner understand the task, or does a miscalculation,

2. If the learner is motivated, or interested, if he/she is under an emotional blockade.

3. Possible errors: Currently for ISAC it is not possible to tell in the beginning of example to tell which fails can be possibly occur. This information would be very useful, because we could know if in a concrete example a learner conducted an expected error or not? However there are more possible way of capturing learners in a very concurrent error (see Dialogue Modes).

We can count or get information under the following conditions, from the following interaction items:

1. **During entering or working with the formula**
   (a) Number of error-patterns detected and the number of correct Formulas
   (b) Number of requested Hints (Fill-Formula) in a concrete example
   (c) Number of requested Automatic Solution
2. **From searching for information, clicking on them**
   (a) Number of viewed Theorems, Pages, Hints
   (b) Number of viewed Explanation
3. **Numbers resulting from the interaction:**
   (a) Number of examples solved, opened
   (b) Number of time invested in solving an example

Other possibilities currently not implemented in ISAC

1. **Set up pages**
   What the teacher or dialogue author thinks is necessary to visit, or recommends to visit. That means during the interaction ISAC could check if the learner did click on these pre-set pages which match error-patterns. If not, the system could just show also the page automatically, the learner has not visited yet.
2. **User Models**
   With user models it opens the possibility for a controlled way of interaction. Maybe teacher, and researcher would be interesting how often does a learner just do one specific error-pattern, and not the whole pre-defined error-patterns. For example, user models, and statistics would enable the following: ISAC could detect errors also the next day, and suggest the learner to go through on selected examples again.

In the ISAC system, there are also other possibilities, a learner can do with the system, but due to the complexity of the problem, in this thesis we wont deal whit these other questions.

In designing counters we have to answer the following questions: What is the beginning value? When will it be increased? when will it be decreased? Do we set it back to original value, if yes, why?
Currently in ISAC Dialogues we have implemented two main counters: One for error-patterns, and an other for a very special case: If the learner is captured in an error-pattern and requests a hint. Further use is described in the rules, see Appendix, and also description of pseudo codes.

error-pattern for session:

1. Name of the Counter: ep_counter
2. After each session it is stored in a database or matrix
3. For every defined error-pattern during the session
4. If an ep is detected, the value increases +1 Beginning value is 0
5. After each session set back to 0
6. System reaction on the counter
   1) if EP detected- hint-page pops up
   2) if EP is bigger than 5 different reaction
   3) if EP is bigger than 10 give easier tasks to solve

Counting number of requesting fill-forms:

1. Name of the Counter: help_counter
2. The value is not stored after a session, only if the learner is caught by an error-pattern, and requests a hint. In this case a fill-form is shown.
3. In case the learner clicks again on the HINT button the value increases +1, and a new fill-form is shown, till there is a fill-form available. Beginning value of the counter is 0.
4. After there is no more fill-form available the value is set back to 0, and automatically the next step will be shown. Also in case the learner closes the window, or steps out of the fill-form requesting system, the value is set back to 0.

4.2.2 A General Collection of Patterns

"Creativity involves breaking out of established patterns in order to look at things in a different way."
Edward de Bono

These patterns were already introduces in an earlier chapter from cognitive science point of view since we found it important to introduce them at that early point of the thesis, and show which pattern will be used in the domain of fractions in ISAC, and why we have chosen this.
In this case for a better overview we also show some concrete examples with concrete numbers (Without a correct solution), for the reason, that with concrete numbers sometimes it is easier to read the examples. This list of error-patterns is a very limited and short list, a lot more error-patterns can be found in the Appendix.

1. Addition and subtraction of fractions:
   (a) $\frac{a}{b} \pm \frac{c}{d} = \frac{a \pm c}{b \pm d}$, example: $\frac{1}{3} \pm \frac{2}{3} = \frac{1 \pm 2}{3 \pm 3}$,
   (b) $a - \frac{b}{c} = \frac{a-b}{c}$, example: $1 - \frac{2}{3} = \frac{-1}{3}$,
2. Multiplication of fractions:
   (a) $\frac{a}{c} \cdot \frac{b}{c} = \frac{a \cdot b}{c}$, example: $\frac{1}{3} \cdot \frac{2}{3} = \frac{1 \cdot 2}{3}$,
   (b) $\frac{c}{a-b} \cdot \frac{d}{b} = \frac{c \cdot d}{a-2b}$, example: $\frac{2}{7-4} \cdot \frac{4}{3} = \frac{2 \cdot 4}{7-2*3}$

(c) $\frac{a}{b} \cdot \frac{c}{b} = \frac{a+c}{b}$, example:$\frac{2}{3} \cdot \frac{4}{5} = \frac{2+4}{3}$

3. Division of fractions:
   (a) $\frac{a}{c} : \frac{b}{c} = \frac{a \cdot b}{c}$, examples: $\frac{1}{3} : \frac{2}{3} = \frac{1 \cdot 2}{3}$,
   (b) $\frac{c}{a-b} : \frac{d}{b} = \frac{c \cdot d}{a-2b}$, example:$\frac{2}{7-4} : \frac{3}{4} = \frac{2 \cdot 3}{7-2*4}$
4. Cancellation of fractions:
   (a) $\frac{a+b}{a} = b$, example: $\frac{2+3}{2} = 3$, $\frac{2+3}{2} = \frac{3}{1}$
   (b) $\frac{a+b}{a+c} = \frac{b}{c}$, example: $\frac{2+3}{2+4} = \frac{3}{4}$
5. Factorisation
   (a) $c + c \cdot b = c \cdot (0 + b)$, example: $5 + 5 \cdot x = 5 \cdot (0 + x)$
6. Special Cases with 0:
   (a) $\frac{b}{c} - \frac{a}{c} = \frac{b-a}{0}$, example: $\frac{3}{x} - \frac{4}{x} = \frac{3-4}{0}$
   (b) $\frac{a}{0} = 0$ or $\frac{a}{0} = a$, example: $\frac{5}{0} = 0$ or $\frac{5}{0} = 5$
   (c) $\frac{a}{a} = 0$, example: $\frac{3}{3} = 0$
7. Special Cases with 1 and -1:
   (a) $\frac{a-b}{b-a} = 1$, example: $\frac{4-5}{5-4} = 1$
   (b) $-\frac{a}{b} = \frac{a}{b}$, example: $-\frac{4}{5} = \frac{4}{5}$
8. Binomial Forms:
   (a) $(a \pm b)^2 = a^2 \pm b^2$, example: $(3 \pm x)^2 = 3^2 \pm x^2$
   (b) $(a - b)^2 = a^2 - b^2$, example: $(3 - x)^2 = 3^2 - x^2$
9. Changing between forms:
   (a) $a\frac{b}{c} = \frac{a+b}{c}$, example: $3\frac{4}{5} = \frac{3+4}{5}$
   (b) $a\frac{b}{c} = \frac{a \cdot b}{c}$, example: $3\frac{4}{5} = \frac{3 \cdot 4}{5}$
10. Priority rules:
    (a) $a - b \cdot c = (a - b) \cdot c$, example: $5 - 6 \cdot 2 = (5 - 6) \cdot 2$
    (b) $a + b \cdot c = (a + b) \cdot c$, example:$5 + 6 \cdot 2 = (5 + 6) \cdot 2$

However, the number of error-patterns are not limited to the above described, and it might depend on the specific learning scenario.

The next sections has been written during a close cooperation with the ISAC developers, and specific parts can also be found on specific page of the ISAC wiki [1], and [2]

[1] http://www.ist.tugraz.at/isac/Rule-based_Dialog
[2] http://www.ist.tugraz.at/isac/Rule-based_Dialog

4.2.3 How to Implement Error-Patterns in ISAC

"Some mathematician has said pleasure lies
not in discovering truth, but in seeking it."
Tolstoy

Error-patterns (EPs) are ISAC's formalization of "misconceptions" in mathematics. They are recognized mechanically by TP-technology and they are a prerequisite for automated generation of adaptive dialogues. For instance, $\frac{1+3}{2+3} = \frac{1}{2}$ is an error well-known to everybody who teaches calculating with fractions. This misconception can modelled as an error-patterns in an ML structure as follows:

```
01 val errpats =
02  [("cancel",
03     [parse_patt thy "(?a + ?b)/?a = ?b",
04      parse_patt thy "(?a + ?b)/?b = ?a",
05      parse_patt thy "(?a + ?b)/(?b + ?c) = ?a / ?c",
06      parse_patt thy "(?a + ?c)/(?b + ?c) = ?a / ?b",
07      parse_patt thy "(?a + ?c)/(?b + ?c) = ?a / ?c",
08      parse_patt thy "(?a + ?b)/(?c + ?a) = ?b / ?c",
09      parse_patt thy "(?a*?b + ?c)/?a = ?b + ?c",
10      parse_patt thy "(?a*?b + ?c)/?b = ?a + ?c",
11      parse_patt thy "(?a + ?b*?c)/?b = ?a + ?c",
12      parse_patt thy "(?a + ?b*?c)/?b = ?a +.  ?b",
13      parse_patt thy "(?a + ?b*?e)/(?c + ?d*?e) = (?a + ?b)/(?c + ?d)"],
14     [@{thm real_times_divide_1_eq}, @{thm real_times_divide_1_eq}])
15   ("addition"), [(*patterns*)], [(*theorems*)]]: errpat list;
```

Error-patterns Definition can be found in Appendix.

In our case we have seen the groups of error-patterns in the previous sub-section. These groups are result of several design decisions.

The decision for the case study appears in the implementation of the error-pattern in ISAC' mathematics engine. The latter is implemented in SML (Milner et al. [87]) (as is the underlying TP Isabelle); the implementation of group addition, in SML is as follows:

```
01 val errpats =
02   [("add-fractions",
03      [parse_patt thy "(?a / ?c + ?b / ?c) = (?a + ?b)/(?c + ?c)",
04       parse_patt thy "(?a / ?b + ?c / ?d) = (?a + ?c)/(?b + ?d)",
05       parse_patt thy "(?a / ?b + ?c / ?d) = (?a * ?c)/(?b * ?d)",

06       parse_patt thy "?a + (?b /?c)= (?a + ?b)/ ?c",
07       parse_patt thy "?a + (?b /?c)= (?a * ?b)/ ?c"],

08      [@{thm rat_add1}, @{thm rat_add2}, @{thm rat_add3}])]: errpat list;
```

According to its definition (See Appendix) an error-pattern is a triple: The first element is the identifier *id*, above
`"add-fractions"`; the second element are a list of *patterns p*, above `[parse_patt thy "(?a / ?c + ?b / ?c) = ..., ...]` (`[]` indicate lists); the third element is a list of rewrite-rules, above `[@thm rat_add1, ...]`. The latter is syntax of Isabelle, addressing the theorem with identifier `rat_add1`. The `?` above indicate a kind of variable which can take other values; this kind of variable allows to detect error-patterns.

In Tab.4.1 on p.54 the error is detected when: the formula $\frac{5 \cdot x}{4 \cdot y} + \frac{3 \cdot x}{4 \cdot y}$ on *line 00* in Tab.4.1 matches the left-hand-side (*lhs*) of the pattern in line `03` above. The match is possible because of the `?`variables: these are mapped to respective values $?a \rightarrow 5 \cdot x, ?c \rightarrow 4 \cdot y, ?b \rightarrow 3 \cdot x$. With this map the right-hand-side (*rhs*) of the pattern in line `03` above is instantiated to $\frac{5 \cdot x + 3 \cdot x}{4 \cdot y + 4 \cdot y}$; and this *rhs* equals the incorrect input $\frac{8 \cdot x}{8 \cdot y}$ on *line 01* in Tab.4.1.

The detection of error-patterns is costly in resources: If error-patterns is detected, the process enforces to check *all* patterns for an input formula, if this formula is *not* related to

any error-pattern. All eight groups of error-patterns listed would comprise several hundred patterns.

The above structure errpats contains two items ”cancel” and ”addition”, ”cancel” is complete and ”addition”, as an example for dialogue authors of how to extend the structure. Each item contains

1. the name, a unique identifier for the error-pattern (for instance, ”cancel”)
2. the patterns, a list of specific patterns for the same general kind of error (for instance, cancelling in a wrong way)
3. the theorems, a list of theorems which have a left-hand-side (lhs) which match the lhs of the patterns.

For instance, these theorems have the same lhs as the ”cancel” error pattern:

```
01 ML {* @{thm frac_eq_eq};
02   @{thm divide_cancel_left};
03   @{thm divide_cancel_right}; *}
04     val it = "?y ~= 0 ==> ?z ~= 0 ==>
       (?x / ?y = ?w / ?z) = (?x * ?z = ?w * ?y)": thm
05     val it = "(?c / ?a = ?c / ?b) = (?c = (0?'a)  ?a = ?b)": thm
06     val it = "(?a / ?c = ?b / ?c) = (?c = (0?'a)  ?a = ?b)": thm
```

The error-patterns are stored in the programs, which guide the calculations within which respective errors are expected. See, for instance, store_met .. ”met_diff_onR” in file Knowledge/Diff.thy.

What happens if no single error-pattern is detected? In this case ISAC works as before: the learner can click on NEXT (providing the next whole step) and AUTO (providing whole automatic solution) button, and search for further information, web content, tactics..etc.. However, in order to ”force” learners to think on their own, one further possibility would be to replace the NEXT whole step with Fill-Forms.

4.3 Hint-Pages

”Logic doesn't apply to the real world.”
Marvin Minsky

One possibility to automatically create dialogue guidance by use of error-patterns is connection with a ”hint-page”. hint-pages can show e.g.: short texts, tips, contain a link..etc.. In most cases for a face-to face learning experience this is completely enough. learners are usually able to correct themselves. For example the following description shows one possibility, how to match these short things with our defined error-patterns. This is of course a design decision, which would be open for any dialogue author to create. and the ability to link a concrete web page to an internet or create complex pages for a learning scenario:

1. **error-pattern: Addition/Subtraction** *You can add fractions if they have the same denominator!*
2. **error-pattern: Multiplication** *Multiplication: Nominator with nominator (denominator with denominator)*
3. **error-pattern: Division** *Division: Multiplication with inversion*

4. **error-pattern: Cancellation** *Something wrong with cancellation! You can cancel if both nominator and denominator are in a form of product*
5. **error-pattern: Factorization** *Something wrong with factorization of fraction!*
6. **error-pattern: Special Case 0** *0 (zero) is a special case! You cannot divide with 0!*
7. **error-pattern: Special Case 1** *Special cases: 1 or -1!*
8. **error-pattern: Binomial Form** *There must be somewhere a binomial form!*
9. **error-pattern: Changing between forms** *Take care, how to change forms!*
10. **error-pattern: Priority Rules** *Take care of priority rule!*

Currently there are the following hint-pages are implemented in the knowledge of ISAC as basic and additional rules, special cases, strategies, and different representations, as it would be required from theoretical point of view. (see Fig.4.3 on p.62).

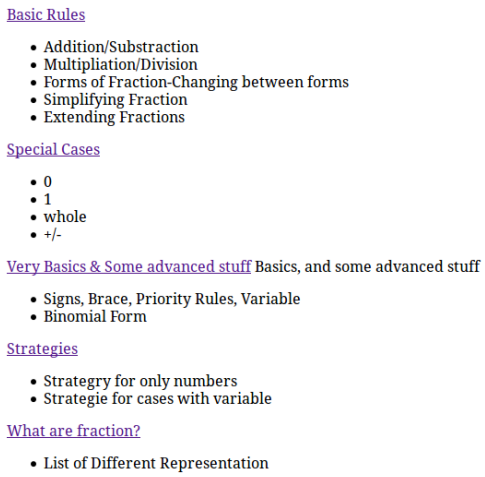

Figure 4.3: Hint-Pages Overview

This contains a very much comprehended list containing correct rules, or short helps as described below (e.g.: in special cases see also Fig.4.4 on p.63):

1. Basic Rules: Contains very simple symbolic explanations how to carry out the following operations on fractions, in order to refresh the necessary knowledge, and collect everything important in a dense form:
   (a) Addition/Subtraction
   E.g.: Correct way of addition or subtraction if they have the same denominator: $\frac{a}{b} + \frac{c}{b} = \frac{a+c}{b}$, in case not, you have to find the common denominator, which in case bd for b and d looks like the following: $\frac{a}{b} + \frac{c}{d} = \frac{a \cdot d}{bd} + \frac{c \cdot b}{bd} = \frac{ad+cb}{bd}$
   (b) Multiplication/Division
   E.g.: Correct way of multiplication: $\frac{a}{b} \cdot \frac{c}{d} = \frac{ac}{bd}$ of $\frac{a}{b} \cdot c = \frac{ac}{b}$ E.g.: Correct way of division: $\frac{a}{b} : \frac{c}{d} = \frac{a}{b} \cdot \frac{d}{c} = \frac{ad}{bc}$
   (c) Forms of Fraction-Changing between forms
   E.g.: Correct way of changing between forms:
   (d) Simplifying Fraction
   E.g.: Correct way of simplification fraction: $\frac{ab}{cb} = \frac{a}{c}$

(e) Extending Fraction
E.g.: Correct way of extending fraction: $\frac{a}{b} = \frac{a \cdot 3}{b \cdot 3} = \frac{a \cdot c}{b \cdot c} = \frac{(a)c}{(b)c}$

2. Special Cases

(a) Zero (0) e.g.: $\frac{a}{c} = 0, if a = c$, $\frac{a}{0} = Notpossible$

(b) One (1) e.g.: $\frac{a}{a} = 1$, $\frac{a}{1} = a$, $\frac{1}{1} = 1$

(c) +/- (negative) e.g.: $-\frac{a}{b} = \frac{-a}{b} = \frac{-1 \cdot a}{b} = -1 \cdot \frac{a}{b}$ or $\frac{-a}{x-b} = -1 \cdot \frac{a}{x-b} = \frac{a}{b-x}$

3. Additional rules

(a) Brackets, Variables, rules

(b) binomial forms e.g.: the first binomial form: $(a+b)^2 = a^2 + 2ab + b^2$, the second binomial form: $(a-b)^2 = a^2 - 2ab + b^2$, and the third binomial form: $(a+b)(a-b) = a^2 - b^2$

4. Strategies

(a) Strategies for examples which contain only numbers

(b) Strategies for examples which contain also variable

5. What are fraction?

(a) List of Different Representation

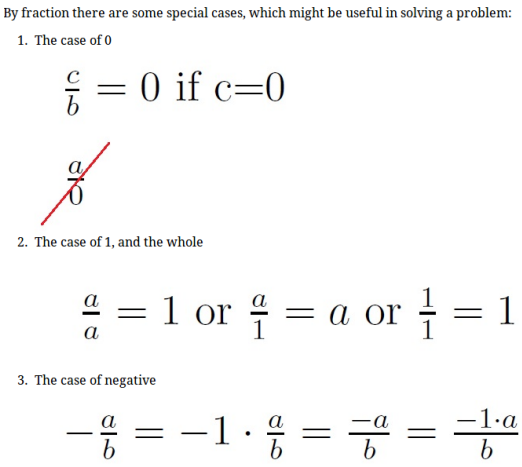

Figure 4.4: Hint-Page for Special Cases

As described above, besides a short hint, hint-pages can be more complex: for example containing tips of the whole domain: and letting learners click on further descriptions, as the hint-pages can show examples of correct solution, as shown in the Fig.4.4, and Fig.4.3, or contain even strategies, which is fitted to the mathematical language level of the learners. These settings depend on the dialogue author, and the teacher who planned the education setting. An example of a strategy is described below:

1. If the task contains numbers and variables.

(a) Try to bring the nominator in the form of a product (A*B, multiplication) with binomial form, or with factoring terms

(b) If there are more fraction: look for the common denominator, and bring fractions to common denominator

(c) Dissolve Brackets in the nominator, and summarize everything that is possible

(d) Try to do the same with the denominator too

(e) Try to simply the fraction

2. If the task contains only numbers

   (a) First change mixed fractions to non-mixed fractions

   (b) If you have more fractions in a brace,
   first you have to bring them to common denominator and make one fraction

   (c) Take care of signs, and carry out multiplication or division

   (d) If possible simplify fraction

How to implement hint-pages into ISAC? Answering this question leads to the question how to extend ISACs knowledge. The process how to add Content in HTML format is detailed described in the Reference, and there is not only a possibility to create HTML "knowledge" sites, but also to add additional examples for problem solving, and this is simple "because this cannot overwrite knowledge prepared in SML data structures"

Hint-pages for error-patterns In case ISAC's math-engine detects an error-pattern, the dialogue might pop up a page, which gives hints about this specific error. This means we have to connect error-patterns to hint-pages. In order to implement this possibility, the file ...hint_pages.properties must get specific entries; for instance, the first two lines implement a hint-page found in the ExampleBrowser's file hierarchy at Examples, Etc, Fractions, Hints, Hint_3.

```
01  HierarchyKey_cancel=Examples,Etc,Fractions,Hints,Hint_3
02  KEStoreKey_cancel=exp_Etc_Frac_Hint_EP3 #
03  HierarchyKey_chain-rule-diff-both=Examples,IsacCore,
04  Calculus,Differentiation,
05  Introduction,chained functions
06  KEStoreKey_chain-rule-diff-both=exp_IsacCore_CDi_chain #
```

This file contains "key=value" pairs, where each error-pattern has two "key=value"pairs:

```
01  the HierarchyKey pointing at the hierarchy in the Examplebrowser
02   key: "HierarchyKey_" + "error-pattern"
03    value: the HierarchyKey: pay attention to have no blanks around ","
04  the KEStoreKey, ISAC's unique identifier for files
05   key: "KEStoreKey_" + "error-pattern"
06   value: the KEStoreKey usually pointing to xmldata/exp
```

Both, the HierarchyKeys and the KEStoreKeys, are found in the respective hierarchy, i.e. in xmmldata/exp/exp_hierarchy.xml. The format of the strings in ...hint_pages.properties is not yet fixed in detail.

4.4 Rewrite Rules

"A mathematician is a device for turning coffee into theorems"
Paul Erdös

Rewrite-rules are the third important element in an error-pattern. We want the learner to do a correct step; however, a step is only correct if we have a rewrite-rule for it. According to the definition of error-pattern rewrite-rules in a error-pattern have a left-hand-side (*lhs*) matching the *lhs* of some patterns. Via the matching patterns an appropriate rewrite-rule is identified; for instance, among the three rewrite-rules in error-pattern "add-fractions" there are two matching the pattern: *rat_add1, rat_add2*:

```
01  ML {* @{thm rat_add};
02    @{thm rat_add1};
03    @{thm rat_add2};
04    @{thm rat_add3}; *}
```

```
01  val it = "?a is_const ==> ?b is_const ==> ?c is_const ==> ?d is_const
02   ==>  ?a / ?c + ?b / ?d = (?a * ?d + ?b * ?c) / (?c * ?d)": thm
03  val it = "?a is_const ==> ?b is_const ==> ?c is_const ==> ?d is_const
04  ==>   ?a / ?c + (?b / ?d+?e)=(?a * ?d + ?b * ?c)/(?d * ?c) + ?e":thm
05  ?a / ?c + ?b / ?c = (?a + ?b) / ?c": thm
06  val it = "?a is_const ==> ?b is_const ==> ?c is_const ==>
07  ?a / ?c + (?b / ?c + ?e) = (?a + ?b) / ?c + ?e": thm
08  val it = "?a is_const ==> ?b is_const ==> ?c is_const ==>
09  ?a / ?c + ?b = (?a + ?b * ?c) / ?c": thm
10  val it = "?a is_const ==> ?b is_const ==> ?c is_const ==>
11  ?a / ?c + (?b + ?e) = (?a + ?b * ?c) / ?c + ?e": thm
12  val it = "?a is_const ==> ?b is_const ==> ?c is_const ==>
13  ?a + ?b / ?c = (?a * ?c + ?b) / ?c": thm
14  val it = "?a is_const ==> ?b is_const ==> ?c is_const ==>
15  ?a + (?b / ?c + ?e) = (?a * ?c + ?b) / ?c + ?e": thm
```

$$
\begin{array}{rcl}
rat\_add1 & :: & \frac{?a}{?c}+\frac{?b}{?c}=\frac{?a+?b}{?c} \\
rat\_add2 & :: & \frac{?a}{?c}+\frac{?b}{?d}=\frac{?a\cdot ?d+?b\cdot ?c}{?c\cdot ?d} \\
rat\_add\_assoc & :: & \frac{?a}{?c}+\frac{?b}{?d}=\frac{?a\cdot ?d+?b\cdot ?c}{?c\cdot ?d} \\
rat\_add1\_assoc & :: & \frac{?a}{?c}+\frac{?b}{?c}=\frac{?a+?b}{?c} \\
rat\_add2 & :: & \frac{?a}{?b}+?c=\frac{?a+?b\cdot ?c}{?b} \\
rat\_add2\_assoc & :: & \frac{?a}{?b}+?c=\frac{?a+?b\cdot ?c}{?b} \\
rat\_add3 & :: & ?a+\frac{?b}{?c}=\frac{?a\cdot ?c+?b}{?c} \\
rat\_add3\_assoc & :: & ?a+\frac{?b}{?c}=\frac{?a\cdot ?c+?b}{?c}
\end{array}
$$

Associativity:
The law of associativity enforces to double the number of theorems because in our case it says: $\frac{5x}{2y}+(\frac{3x}{2y}+1)=(\frac{5x}{2y}+\frac{3x}{2y})+1$ and in order to detect the error-pattern of the example in the representation of the RHS we need theorems add_assoc, add_assoc1..etc.).

4.5 Fill-Forms

"Mathematics is as much an aspect of culture
as it is a collection of algorithms."
Carl Boyer

Fill-patterns are associated with rewrite-rules addressed in error-patterns. Fill-patterns contain *fill-in-patterns* with space-holders and the prototype uses fill-patterns to provide services for adaptive user-guidance. These are left blank when instantiated: for instance, in Tab.4.1 line 00 the formula $\frac{5 \cdot x}{4 \cdot y} + \frac{3 \cdot x}{4 \cdot y}$ matches line `03` of the error-pattern, as already mentioned. From the error-pattern's rewrite-rules *rat_add1* is selected, and *rat_add1*'s fill-patterns are implemented in SML as follows: (However, there could be generated automatically, these are set manually), and with each new request on help, the number of space-holders are decreasing, providing less help in the beginning, and more in the end. In our case the following list (because of space-issues the list does not contain all possibilities, e.g. subtraction) shows the fill-patterns for the error-pattern *add-fractions*. It was important to distinguish between $\frac{a}{b} + \frac{c}{d}$ and $\frac{a}{b} + \frac{c}{d}$, because they require different fill-patterns

1. For the running example $\frac{a}{c} + \frac{b}{c} = \frac{a+b}{2c}$, $\frac{a}{c} + \frac{b}{c} = \frac{+..}{..}$, or $= \frac{..+..}{c}$, or $= \frac{a+..}{c}$
2. For $\frac{a}{b} + \frac{c}{d} = \frac{a+c}{b+d}$, $\frac{a \cdot .. + .. \cdot ....}{..\cdot ..}$, or $= \frac{a \cdot d + .. \cdot ....}{b \cdot ..}$, or $= \frac{a \cdot .. + .. \cdot ....}{b \cdot d}$..etc.
3. For $a + \frac{b}{c} = \frac{a+b}{c}$, or $a + \frac{b}{c} = \frac{..\cdot..+..}{c}$,or $= \frac{..\cdot c+..}{c}$, or $= \frac{a \cdot c+b}{..}$

These fill-patterns are defined in ML code similar to rewrite-rules. Because of space issues we show only the ML code for the running example from the Table 4.1:

```
01 val fillpat =
02    [("fill-addition-first1",
03      parse_patt @{theory Rational} "
       (?a / ?c + ?b / ?c) = ( _ + _ ) / (  _  )",
04      ["add-fractions"]),
05     ("fill-addition-first2",
06      parse_patt @{theory Rational}
        "(?a / ?c + ?b / ?d) = ( _ + _ ) / (?c) ",
07      ["add-fractions"]),
08     ("fill-addition-first3",
09      parse_patt @{theory Rational}
        "(?a / ?c + ?b / ?d) = (?a +  _) / (?c ",
10      ["add-fractions"]),
11     ...
12    ]: fillpat;
```

```
01  ("fill-addition-second1",
02   parse_patt @{theory Rational} "?a + (?b /?c) = ( _ * _ + _ )/ ?c",
03    ["addition-of-fraction" ]),
04   ("fill-addition-second3",
05    parse_patt @{theory Rational} "?a + (?b /?c) = ( _ * ?c + _ )/ ?c",
06      ["addition-of-fraction"]),
07   ("fill-addition-second5",
08    parse_patt @{theory Rational} "?a + (?b /?c) = ( ?a *?c + ?b )/ _",
09    ["addition-of-fraction"]),
```

Here exactly those fill-in-patterns are shown, which generated the lines `03..05` in Tab.4.1; the place-holders are represented as _. It is imaginable to create the place-holders automatically. It is noted, that the third element in a fill-pattern's triple,
e.g. `"add-fractions"` above, are provided for future services for the dialogue not yet implemented.

Other examples for fill-pattern

1. $\frac{a}{c} \cdot \frac{b}{c} = \frac{a \cdot b}{c}$, $\frac{c}{a-b} \cdot \frac{d}{b} = \frac{c \cdot d}{a-2b}$ or $\frac{a}{b} \cdot \frac{c}{b} = \frac{a+c}{b}$ possible answers:
   $\frac{a}{c} \cdot \frac{b}{c} = \frac{\cdot\cdot \cdot \cdot\cdot}{\cdot\cdot \cdot\cdot}$, $\frac{a}{c} \cdot \frac{b}{c} = \frac{a \cdot b}{\cdot\cdot \cdot\cdot}$, $\frac{a}{c} \cdot \frac{b}{c} = \frac{\cdot\cdot \cdot\cdot}{c \cdot c}$
2. $a \cdot \frac{b}{c} = \frac{a \cdot b}{a \cdot c}$, possible answers:
   $a \cdot \frac{b}{c} = \frac{\cdot\cdot \cdot\cdot}{\cdot\cdot}$, $a \cdot \frac{b}{c} = \frac{\cdot\cdot \cdot\cdot}{c}$, $a \cdot \frac{b}{c} = \frac{a \cdot\cdot}{c}$
3. $\frac{a}{b} : \frac{c}{d} = \frac{a : c}{b : c}$ possible answers:
   $\frac{a}{b} : \frac{c}{d} = \frac{\cdot\cdot}{\cdot\cdot} \cdot \frac{\cdot\cdot}{\cdot\cdot}$, $\frac{a}{b} : \frac{c}{d} = \frac{a}{b} \cdot \frac{d}{\cdot\cdot}$, $\frac{a}{b} : \frac{c}{d} = \frac{a}{\cdot\cdot} \cdot \frac{d}{\cdot\cdot}$, $\frac{a}{b} : \frac{c}{d} = \frac{a}{\cdot\cdot} \cdot \frac{d}{c}$

One group of error-pattern might contain several different patterns, which request different fill-forms. Grouping error-patterns has the advantage, for example in the case of hint-pages,that we can give the same hint-page for a branch of groups of patterns, but this not influences the capacity that the correct fill-form and fill-pattern will be shown to the learner in case it is requested.

4.6 Dialogue Modes

"It is with children that we have the best chance
of studying the development of logical knowledge,
mathematical knowledge, physical knowledge, and so forth."
Jean Piaget

Possible Problems, limits of the error-pattern detection: There are some steps, where the machine has no chance to detect a error. Often happens, that learners just jump many steps, and write a very short solution. Without recounting the example step by step, in this cases also a human teacher would make a hard time to detect where the error occurred. With some experiences, teacher might develop an intuition. This might also be supported in ISAC with the help of statistical empirical data. Because that might be in the beginning of the implementation a lower possibility of detecting errors, we introduce the so called dialogue modes (DM) Dialogue Mode comprises settings which are the same for all learners at each level etc. Presently there are two modes.

1. **normal-mode**: the dialogue checks input for EPs; if an EP is found, normal handling is done: EP_counter, etc.
2. **check-EPs-mode**: forces the individual learner to formulas, where only one EP can happen. This mode is switched on by error_patterns_to_check
3. **Guiding Through an Example**: allows steps which

The dialogue might have recorded, that this individual learner has already activated the "cancel" error-pattern several times. On the other hand, fill patterns can be used even if there is no related error-pattern; for the latter case there is a special identifier "no-error-pattern". In ideal case "Dialogue Modes" would help to choose the best teaching and learning strategy, based on the collected data as: background information (e.g.: what is the goal of the learner, how much time is available), and would be also able to detect the level of knowledge, skills.

The rule-based dialogue engine has just been implemented (Kienleitner [64], Kober [68]), the Java code switching between rule-sets was a stub at the beginning of this thesis; this stub has been extended in parallel to the development of the rule-sets in this thesis.

This section is a first record of guidelines for dialogue authors; it has been written during cooperation with the ISAC developers, this section is mirrored on a specific page of the ISAC wiki[3], the subsequent paragraph "architecture of ISAC's Dialog Guide" is at *http://www.ist.tugraz.at/isac/index.php/Dialog_Architecture*.

[3] http://www.ist.tugraz.at/isac/index.php/Guidelines_for_Dialog_Authoring

The architecture of ISAC's Dialog Guide separates the dialogues clearly from the other components in ISAC. This is shown in Fig.4.6 on p.68. The relevant components are

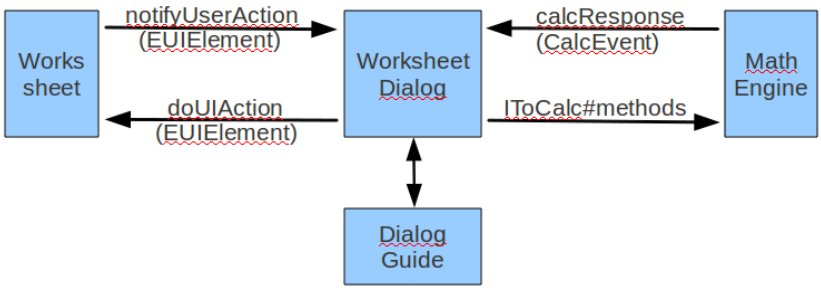

Figure 4.5: The WorksheetDialog with Connections.

WorksheetDialog manages the dialogue for exactly one Worksheet

Worksheet "Worksheet displays exactly one calculation, which might consist of sub-problems. Within the Worksheet also the specification phase is done in a separate window."

MathEngine "MathEngine does all calculations requested by the learner (via Worksheet) and by the WorksheetDialog. The latter, for instance, wants to check which error-patterns are related to the current step of calculation. This check involves "matching" of formulas, which can only be done by the MathEngine."

DialogGuide "DialogGuide serves all dialogs, in particular the WorksheetDialogs and the dialogs for the browsers for examples, theories, problems and methods. The Dialog-Guide manages the hint-pages, several counters etc. "

4.7 Implementation of Dialogue-Rules

"Reserve your right to think,
for even to think wrongly is better
than not to think at all."
Hypatia of Alexandria

Error-patterns and fill-patterns are implemented in ISAC's *mathematics-engine (ME)* which is appropriate because they involve rewriting, a technique from Computer Mathematics. Dialogue-rules are implemented in the Java-based part of ISAC, where the dialogue component resides, mediating between the learner working on the *Workhseet (WS)* and the ME, so it is called *WorksheetDialog (WD)* [4]. The software architecture has been designed such, that via dialogues users (learners and teachers) can watch the Worksheets of others, or even can input to them. These features will be exploited as soon as basic user-guidance is implemented in the *DialogGuide (DG)*.

The behaviour of ISAC's dialogue is determined by *dialogue-rules* interpreted by Drools Expert (Amador [3]), a knowledge-based expert system. These rules are triggered by events addressing the WD on top from left and right, and the rules in the WD can determine actions directed towards left (WS) and right (ME) ([35]).

1. *notifyUserAction* notifies the WorksheetDialog about an action of the user on the Worksheet by an **EUIElement**. For instance, in `rule 2` the trigger is `EUIElement`

[4]The Workhseet on the front-end is accompanied by further windows accepting user-input; these windows have their own dialogues and are out of scope of this thesis

==
`UI_SOLVE_HELP_ENTERING_FORMULA` indicating that HELP has been pushed.

2. *doUIAction* notifies the Worksheet about what to display. This methods uses the same **EUIElement** as (1.) compound with appropriate data (i.e. the actions between WS (learner) and WD (the 'system') are symmetrical). The rules in the running example do not involve such notifications, because fill-in-forms are passed to the WS by CalcChanged without intervention of the WD.

3. *calcResponse* notifies the WorksheetDialog about the result of the MathEngine for the last request for calculation from the WorksheetDialog by a **CalcEvent**. This can be of 2 kinds:

   (a) **CalcChanged** notifies about a successful step of calculation. This case occurs when an error-situation is quit by a correct input which "changes the calculation" by adding a new line. Since the dialogue cannot interpret formulas, CalcChanged passes formulas to the WS by-passing the dialogue. Thus the rules concerning error-patterns are not triggered by CalcChanged, but only by CalcMessage, see `rule 1` ([35]).

   (b) **CalcMessage** notifies about a error of the MathEngine trying to do the last requested step of calculation. For instance, in `rule 1` on line `03` an error-pattern is reported together with an `errorID`, in the running example *add-fractions*.

4. **IToCalc#methods** request services from the MathEngine, which is addressed via interface IToCalc, an abstraction of the calculation on the Worksheet with respective positions, formulas and tactics/rules. For instance, `rule 2` sets the action `ME#requestFillformula(err_patt, fill_patt).`

5. Services from *DialogGuide (DG)* are access to hint-pages, update of counters etc. For instance, `rule 1` contains `DG#showHintPage (err_patt_).`

The executable format of Drools' rules involves distracting technicalities, so the rules below are written in a pseudo-code which is both, comprehensible for a non-programmer (like a dialogue-author) and succinct enough for a programmer translating them to executable code [5].

```
01 rule "1: show a hint-page if an error-pattern is detected"
02   when
03     CalcMessage == "error-pattern#errorID#"
04   then
05     error_pattern_ = CalcMessage.getErrorPattern()
06     error_patterns_.add(error_pattern_)
07     DG#showHintPage (err_patt_)
08     DG#EP_counter (err_patt_) ++
09     help_counter_ = 0
10     WS#addHelpButton()
11     WS#removeNextAndAutoButtonForWorksheet()
12   end
```

Besides giving hints it is also important to involve learners actively in the learning process, and let them think. We don't let them just request the next correct step or formula (see line `11` above), but make them struggle a bit. However, line `10` adds a HELP button (see Fig.4.2) for requesting fill-forms.

```
01 rule "2: from ME request fill-patterns"
```

[5] The *ISAC*-project expects to develop a dialogue-language which copes with the need of both, authors and programmers, within one single format

```
02   when
03     EUIElement == UI_SOLVE_HELP_ENTERING_FORMULA
04   then
05     fill_patts_ = ME#findFillpatterns(err_patt_)
06     fill_patt = DG#selectFillPattern(fill_patts_, help_counter_)
07     ME#requestFillformula(err_patt, fill_patt)
08     help_counter_ ++
09   end
```

`rule 3` repeats the request for a fill-form. The DG knows which one to select from the `help_counter_`, see line `07` below. Finally, `rule 4` proposes the tactics, i.e. the rule, leading to the correct next formula.

```
01 rule "3: for incorrect fill-form request other fill-form"
02   when
03     CalcMessage == "fill-form incorrect"
04     && help_counter < length(fill_patts_)
05   then
06     fill_patt =
07       DG#selectFillPattern(fill_patts_, help_counter_)
09     help_counter ++
10   end
```

Given the 37 dialogue-rules handling interactions with NEXT, AUTO, input of a formula, etc ([35]). The above 4 dialogue-rules are sufficient to extend ISAC's dialogue behaviour in the decisive way described in this thesis.

```
01 rule "4: fill-forms did not help, show rule"
02   when
03     ((CalcMessage)calc_event).getText() == "fill-form incorrect"
04     && help_counter >= fill_pats_no
05   then
06     calc_tree.fetchProposedTactic();
07   end
```

The dynamic behaviour of the four rules is shown in Fig.4.6 on p.70. The reader may note, that the four rules are neither specific for addition, nor fractions — they are general such that they handle all domains of mathematics.

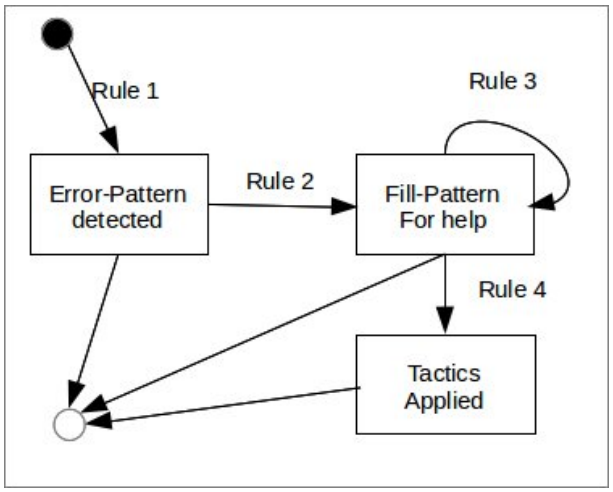

Figure 4.6: State-Diagram of the Dialogue-Rules

These rules also involve some counters (helping to get more information from the dialogue process):

1. Counter for the fill-pattern, and counter for the error-pattern. The first counter for the fill-pattern is important because we want to implement, that by requesting the first fill-pattern the learner gets relatively small help, and if they are requesting more help, they get more information.
2. The second one, the counter for the error-pattern is important, because: if the learner conducts the same error-pattern during the learning session, he/she cannot get always the same reaction. The computer should provide a variety of answers. After the first 1-2 times, detecting the same error-pattern, the hint-page for example will not appear again, but some different reaction.

For creating other own rules dialogue authors have to look up directly in the java-file. These contain further possibilities (partly described in this thesis), and the following list tries to show how these might look like:

1. Show or request tactics in the following situation
   (a) What tactics would the mathematics engine apply UI_SOLVE_GET_PROPOSED_TACTIC
   (b) List of tactics to the active formula UI_SOLVE_GET_APPLICABLE_TACTICS
   (c) Set the tactic applied to the current formula UI_SOLVE_SET_NEXT_TACTIC
   (d) Show the tactics applicable for this formula UI_SOLVE_SHOW_APPLICABLE_TACTICS
   (e) Which tactic was applied? UI_SOLVE_TACTIC_APPLIED
2. Show theorem, Show browser, Show intermediate step
3. Calculate till a sub-problem is solved UI_SOLVE_CALCULATE_SUBPROBLEM
4. Buttons in ISAC: Help Button (UI_SOLVE_HELP_ENTERING_FORMULA), Auto, Next,
5. Formulas
   (a) Request editing the currently active formula UI_SOLVE_EDIT_ACTIVE_FORMULA
   (b) Currently active formula is finished UI_SOLVE_EDIT_ACTIVE_FORMULA_COMPLETE
   (c) New formula after the currently active formula UI_SOLVE_APPEND_USER_FORMULA
   (d) Make the reference formula the currently active formula
   UI_SOLVE_MOVE_ACTIVE_FORMULA

Chapter 5

Embedding $\mathcal{ISAC}$ into Learning Scenarios

"I came to the . . . open gate of mathematics."
M.C.Escher

What would be a concrete example of the benefit that could be created if cognitive science is applied to $\mathcal{ISAC}$? This chapter will show a possibility for a dialogue guidance, and implementation of a concrete scenario.

Staying with the example of the fraction, we would like to compare real life examples with the possibilities of $\mathcal{ISAC}$.

5.1 Examples of Errors in the Domain of Algebra

"Perfect numbers like perfect men are very rare."
Descartes

A question is, if the above described errors are elaborate and inclusive enough, to detect errors from learners. In this paragraph we will show some examples of errors conducted by a learner age fifteen (collected during observation in a case study). These examples were collected on a sheet of paper, but for a better readability, we will show the errors in the following way: Examples will be shown in a frame followed by the learner's solution, followed again by the correct solution, and a short explanation.

Example 1: $\boxed{\frac{x-\frac{1}{x}}{\frac{(1+x)}{x}}} = \frac{\frac{x^2-1}{x}}{\frac{1+x}{x}} = \frac{x^2-1}{x} * \frac{1+\not{x}}{\not{x}}$

Correct solution: $\frac{x-\frac{1}{x}}{\frac{(1+x)}{x}} = \frac{\frac{x^2-1}{x}}{\frac{1+x}{x}} = \frac{x^2-1}{x} * \frac{x}{1+x} = \frac{\cancel{(x+1)}(x-1)}{\not{x}} * \frac{\not{x}}{\cancel{1+x}} = x-1$

As we can see, the last step step contains errors:

1. **Error in Division:** One can divide fractions if we multiply with the reciprocal of the divisor. In the above example, the multiplication was "correct", but the reciprocal of the fraction was obviously forgotten. This pattern can be found in the group: "Division", and it would be possible for $\mathcal{ISAC}$ to detect it.

2. **Error in Cancellation:** $\frac{1+\not{x}}{\not{x}}$ would only be correct if instead of + there would be a multiplication sign. This pattern can be found in our group: "Cancellation"

3. **Recognizing binomial form:** It is obvious, that in order to solve the example, the learner should have realised that $x^2 - 1 = (x+1)(x-1)$. This pattern can be found in our group: "Binomial Form"

The above example indicates a difficulty: the difficulty to detect error-patterns, if several of occur in one step. In such cases it might be possible for a teacher, what are these errors, and what is their nature. In the above case ISAC would not detect both errors at the same time, only if the learner was be asked to go on step by step. However, error-pattern "Division" could be detected, and the system could answer with $\frac{x^2-1}{x} * \frac{x}{\dots}$, for instance.

Example 2: This example was part of a task, where the denominator was to be simplified: $\boxed{\frac{2(x-1)*x}{(x-1)*x}}$=$\frac{2x-2x}{(x-1)*x}$

Correct solution: $\frac{2(x-1)*x}{(x-1)*x}$=$\frac{(2x-2)x}{(x-1)*x} = \frac{2x^2-2x}{(x-1)*x}$

This is quite a common error in term multiplications and erasing brackets. This could be an example of our error-pattern, algebraic sign, and in case of a request for a help ISAC could answer with the following fill-form: $\frac{2(x-1)*x}{(x-1)*x} = \frac{(\dots-\dots)*x}{(x-1)*x}$

Example 3: $\boxed{\frac{1}{a+1} + \frac{1}{a-1}}$=$\frac{1\cancel{(a-1)}}{(a+1)\cancel{(a-1)}} + \frac{1\cancel{(a+1)}}{\cancel{(a+1)}(a-1)} = \frac{1}{(a+b)(a-b)}$ and

$\boxed{\frac{100}{a} + \frac{10}{b}}$=$\frac{110}{ab}$

Correct solution: $\frac{1}{a+1} + \frac{1}{a-1} = \frac{1(a-1)}{(a+1)(a-1)} + \frac{1(a+1)}{(a+1)(a-1)} = \frac{a+1+a-1}{(a+b)(a-b)} = \frac{2a}{(a+b)(a-b)}$

Correct solution: $\frac{100}{a} + \frac{10}{b} = \frac{100b}{ab} + \frac{10a}{ba} = \frac{100b+10a}{ab}$

This is a great example for our error-pattern "Addition". However, also an example for how hard it might be to understand where error-patterns originate from. In the first case of the Example 3 we cannot know if it is simply an error in the nominator $1 + 1 = 1$, or the pattern: $\frac{a}{b} + \frac{a}{c} = \frac{a*a}{bc}$. Nor a human teacher without further interaction with the learner, nor can ISAC make a decision here. But in this case ISAC would recognise it as the second pattern: $\frac{a}{b} + \frac{a}{c} = \frac{a*a}{bc}$, and in a case of requesting help, besides a hint-page, ISAC could give the following help: $\frac{..+..}{(a+b)(a-b)}$

Example 4: $\boxed{36x^5 - 12x^3}$=$12(2x^5 - x^3)$

Correct solution: $36x^5 - 12x^3 = 12(3x^5 - x^3) = 12x^3(x^2 - 1) = 12x^3(x + 1)(x - 1)$

Sometimes learners dont know when to end an example, as mentioned in the previous chapter as well. In this case ISAC can give the following help, and fill-form: $12(2x^5-x^3) = 12 * ..(.. - 1)$

Example 5: $\boxed{\frac{1}{a+1} * \frac{1}{a-1}}$=$\frac{1\cancel{(a-1)}}{(a+1)\cancel{(a-1)}} * \frac{1\cancel{(a+1)}}{(a-1)\cancel{(a+1)}} = \frac{1}{(a+1)(a-1)}$

Correct solution: $\frac{1}{a+1} * \frac{1}{a-1} = \frac{1}{(a+1)(a-1)}$

This example is very interesting to show how the learning process takes places: sometimes applying a "wrong" rule (In this case there was no need of extending the fractions to apply the multiplication) might lead to good solutions. This shows that in this case extending and cancelling fractions worked out well, but for some reasons there was a failure in multiplication .The same learner can make an error in the next example forgetting how to extend fractions.

Example 6: $\boxed{-5a - (-7)}$=$2a$

Correct solution: $-5a - (-7) = -5a + 7$

There are some error-patterns that are not mentioned in this thesis, as the Example 6. This thesis shows the possibility of error-patterns but does not have the goal to establish all subdomains connected to fractions.

There could be a lot more examples, but not all can be connected to or be found in theoretical background. From a cognitive science point of view it would be very interesting

to understand where these errors originate from, what is their underlying structure, and implement this knowledge into an educational software.

The conclusion is, that there is a high possibility, that learners conduct several errors at the same time. In the above show cases it was rather easy to see where the error was, but if the learner commits several mistakes at the same time, it becomes rather complicated also for a teacher him/herself to detect these errors. It would be an interesting question to ask: which errors happen together, or with higher probability at the same time. At present it seems best to lure students into proceeding in small steps where multiple errors do not occur. How to give help in a stepwise solution is described subsequently.

5.2 Guiding Through an Example

"Some people believe in imaginary friends.
I believe in imaginary numbers."
R.M. ArceJaeger

Simplifying fractions follows algorithms which allow support (by Lucas-Interpretation) for step-wise approaching a result similar to paper and pencil work. Each step comprises application of one or more rules, so we can take different knowledge levels of the learners into consideration.

In curricula simplification of fractions often starts with addition of fractions (see examples in the Appendix), because it is more complex than simplification of *one* fraction, secondly addition requires to compute the common denominator. Let's assume, that in this concrete chapter the goal is to add/subtract two/more fractions and to simplify them (e.g.: $\frac{a+1}{a^2-a} + \frac{b+1}{ab-b}$). This needs specific background knowledge, and learners might be on different knowledge levels.

Using the knowledge from cognitive science described in the above chapters we can design the interaction basically for two groups, however we should assume, they both already knew the basics. One group requiring a different abstraction level and closer connection to real-world examples. That means: blending the task to a different representation (for the other group we can use the common mathematical language). This abstraction level close to real-life is supported by (Koedinger et al. [70]), he called it "storyfing" and stated: at some level of knowledge a kind of "storyfing" is needed, and later on we can move on to more mathematical abstraction. We will also take this notion and use it in the further steps. Vygotskian theory says that learning proceeds from the concrete to the abstract (Rogoff et al. [119]).

The following mathematical knowledge is necessary to successfully add and subtract fractions. Basically they are different parts of mathematics and are considered separately (sometimes even in different years), and this task of adding and subtracting fractions requires learners to integrate all this knowledge and to apply rules in the right order:

1. Factoring terms
   This means the learner has to be able to:
   (a) cancel a number from the term
   (b) cancel a single variable or a whole sub-term
   (c) recognize a binomial form
2. simplify terms or fractions (which involves knowing when a task ends)
3. create common denominators (LCM) of algebraic terms
4. expand fractions according to a desired denominator (In this task learners conduct often errors with brackets)

5. operate with the basic rules of algebra (associativity, commutativity, etc)
   The process described above usually involves the following steps:
   (a) remove brackets (taking into account precedence questions, taking care of minus signs before brackets)
   (b) carry out addition/subtracting terms (for that one has to understand which terms can be added, which not)
   (c) factor terms if possible, where we come back again to a same process.
   (d) know where to put the brackets, what to do with minus signs, etc.

During a personal face to face session with the learner it is possible to tell the necessary strategy et each point. In this situation the teacher can identify, with respect to which of the background knowledge described above the learner struggles, what is missing, etc..

The problem with the procedure described above is the unlikeness of finding an underlying error-pattern during normal use of ISAC. To increase the possibility of finding an error-pattern, and knowing what exactly the learner tries to do, ISAC could guide him/her trough an example, and capture at single moments, where a limited amount of error-patterns might occur and where we could benefit from the fill-forms. With the technological background, its fine-grained detection of error-patterns, its universal usable fill-forms, ISAC would bring the possibility to capture learners in a very single moment, and look not only on a level of single error-pattern where the problems are, but also identify the crucial processes e.g: that most errors occur during factorisation. That way we could get much closer to the source of the errors than with human observation and guessing.

The quite simple following strategy (or algorithm) for adding and simplifying fractions (Multiplication and division would need a different strategy)-also experienced by the user can be implemented within ISAC (If the example only involves simplification of a single fraction, the strategy is much easier. It contains just: Factorization, and simplification.– First and second step in the following list.)

1. First step is to factorising all nominators and denominators, *Usually the problem is that the learners forget, or do not know how to factorise an algebraic term, which is a key in finding the LCM.(knowledge needed: being able to factorise an algebraic term is key*).
2. and look if one can simplify any of the fractions *(knowledge needed: simplifying fraction)*, because mathematics goes with "the usually the simpler the better".
3. Look for common denominator, if they are unlike. *Finding the common denominator might be the challenge because it involves the concept of LCM(= Smallest/least common multiple or common denominator)*
   (Knowledge needed: finding LCM) Learners usually struggle with the following problems here:
   (a) Dont remember that they can add fractions only if they have the same denominator
   (b) Dont understand the concept of LCM, especially if the expression involves algebraic terms
4. Rewrite the fractions as equivalent fractions with the LCM as the denominator, and find the expansion factor for that *(Knowledge needed: Expanding Fraction, recognizing what is missing from the nominator)*
5. Successfully carry out the addition *(knowledge needed: manipulating terms)*
6. Factorizing nominator and denominator *(knowledge needed: factorization)*

7. Simplify again if needed *(knowledge needed: Simplification, and to know when to end an example)*

Usual tips for preventing a lot of errors and miscalculation during the whole process are:

1. Look if you have all the brackets you need
2. Look at minus signs
3. Look again if each number, and sign is correct, and the same as in the previous lines.

Another problem is, that learners dont know the line up of a single steps they have to follow. Basically it is always the same, however in the learning process they might get mixed up.

This strategy could be successfully implemented into ISAC and be considered in the examples above. Once again we have to choose an approach according to the two levels described in this chapter: One with concrete mathematical language, and the second for a more, or less abstract, and blended way: as a "game". For the "game" we will create the following rules in ISAC, and the next table shows the rules on both "storyfied" and normal mathematical language. First we would like to show the concrete plan of the interaction with normal mathematical language, than introduce the "game" which uses a different abstraction level, and different language. Basically both scenarios are logical identical, and guide the learner through the problems as described above.

Concrete example of interaction: How to read the tables? Each table has four rows. The first row shows the actual example on the worksheet (in the table called: display), the second line is the guiding text which can be seen on the screen. The third line shows the learners possible interaction (Because this guided mode is on purpose limited in interaction, these are the steps we could allow the learners to do). The last line shows ISACs possible reaction on the learners input. In the background ISACs automatic error detection is still on, and the fact that we limit the learner in his/her interaction makes it possible to exclude multiple errors happening at the same time, and makes also possible to detect the error-patterns, and to identify at which step, and/or process do these errors occur.

1. Step: Here we can check if the learner is able to recognize if fractions have the same denominator or not

Example on the display	$\frac{a+1}{a^2-a} + \frac{b+1}{ab-b}$
Text on the display	Do they have the same denominator?
learners reaction	Click on YES or NO
ISACs reaction	right/or no answer (in case no:Hint)

2. Step: We are in the process of factorization

Example on the display	$\frac{a+1}{....} + \frac{b+1}{ab-b}$
Text on the display	Factorize the first denominator
learners reaction	Correct, or request Help (Fill-Form)
ISACs reaction	Fill-Form: $a(.. - ..)$

3. Step: Still in the process of factorization

Example on the display	$\frac{a+1}{a(a-1)} + \frac{b+1}{..}$
Text on the display	Factorize the second denominator
Possible reaction of the learner	Correct, or request Help (Fill-Form)
Possible reaction of the ISAC	E.g.: Fill-Form: $..(a-1)$

4. Step: If all denominators are correctly factorized, we can go the the process of creating the LCM

Example on the display	$\frac{(a+1)\cdot(..)}{..} + \frac{(b+1)\cdot(..)}{..}$
Text on the display	What is the common denominator?
Possible reaction of the learner	Fill in or Request Fill-Form/hint-page
Possible reaction of the *ISAC*	hint-page as table seen above:

In this case it is only possible to write in the part of the denominator, and the requested hint-page is the same as the table seen above, only with missing common denominator, and an explanation what it is. If needed with highlighting all common parts. (As pattern matching)

5. Step: If the common denominator was found, it is possible to step to expansion

Example on the display	$\frac{(a+1)\cdot(..)}{CD} + \frac{..}{CD}$
Text on the display	Find the expanding factor for the 1. fractions!
Possible reaction of the learner	Correct, or request interactive Help Page
Possible reaction of the *ISAC*	Produce the interactive page

The interactive page would refer separately for the actual fraction, and highlight the corresponding patterns and parts.

Name	Denominator	Factorized den.	Expanding factor
1. Fraction	$a^2 - a =$	$a(a-1)$	b
2. Fraction	$ab - b =$	$b(a-1)$	a
Common Denominator	..	$ab(a-1)$	...

6. Step: If we have the common denominator, and first expanding factor correctly (Process of Expansion)

Example on the display	$\frac{(a+1)\cdot(b)}{CD} + \frac{(b+1)\cdot(..)}{CD}$
Text on the display	Find the expanding factor for the 2. fractions!
Possible reaction of the learner	Correct, or request an interactive Help Page
Possible reaction of the *ISAC*	Produce for each pairs interactive page

7. Step: After all expanding factors were found one can carry out addition or subtraction

Example on the display	$\frac{....}{ab(a-1)}$
Text on the display	Add/Subtract the fraction
Possible reaction of the learner	Correct, or request interactive Help Page
Possible reaction of the *ISAC*	Hint can show the first part or (auto) solve

The above described strategy used common mathematical language, the next describes the "blended" version which can be more suitable for some learners.

Description of the "Game" :
Overall Goal is:*make two or more fractions "play" together.*
For example: $\frac{a+1}{a^2-a} + \frac{b+1}{ab-b}$ want to "play together", but this requires some special rules, and a "preparation" from the learner.
(This case of playing is the equivalent of addition or subtraction. For multiplication and division we would need a different rules, and different games. Basically this game builds up on already existing knowledge.)
General Rules are: Two or more fractions "can play together" if they have the same common "rule-set" Each fraction has its own rule-set, which can be found in the "lower" part of the fraction. Sometimes rules can appear complicated, and are built from so called "basic rules", or "basic building elements". The first task to always find which are the basic rules. (Through factorization).

For example the following fraction: $\frac{b+1}{ab-b}$, has its own rule: $ab - b$, which contains the basic rules or basic building elements: b and $a - 1$, since: $ab - b = b(a - 1)$. (In this step of the game we should assume that the learner knows the way of factorisation, or *ISAC* will

provide help with the fill-forms described below as well).

Generally keeping things simple is always a good way, so if at all possible, we work with "basic rules" For two fractions to be able to play together they have to have the same rule-sets, as already mentioned. First before playing "they have to learn" multiplying the whole above part of the fractions with the not yet known rule. The common rule set: is all basic rules (but once, only once!) which can be found in all fraction.

The following table, usually used also in mathematics teaching, shows for our example $\frac{a+1}{a^2-a} + \frac{b+1}{ab-b}$ for each fractions separately which own rules, own basic rules they have, and which rules does the learner have to learn.

Name	Own ruleset	=known basic rules	Rules to learn
1. Fraction	$a^2 - a =$	$a(a-1)$	b
2. Fraction	$ab - b =$	$b(a-1)$	a
Common Rules	..	$ab(a-1)$	...

As we can see the first fraction has the rule: a and $a-1$, the second b and $a-1$, so both know already $a-1$, the first miss b the second:a, which we have to teach them first before they can play. The first fraction with b, the second one with a:

$$\frac{(a+1)\cdot b}{(a^2-a)\cdot b} + \frac{(b+1)\cdot a}{(ab-b)\cdot a}$$

Now, when both know the same rules they are able to "play" together as shown:

$$\frac{(a+1)\cdot b-(b+1)\cdot a}{(ab(a-1)}$$

From this point on we refer to the knowledge of addition and subtraction, and multiplication within terms. This kind of "game" is good for learners who struggle with the notion of the common denominator, as well as with the expansion of fractions containing terms. This is equivalent to the use of normal mathematical language, as adding..etc.

The whole process could be connected to a rewarding point collecting system. This implemented guided design would be an example to represent the possibilities of ISAC's general error detection capabilities as well.

5.3 Proposals for Extending ISAC

"To understand the things that are at our door
is the best preparation for understanding those that lie beyond."
Alexandria of Hypatia

The need and relevance to design technology-integrated environments to support mathematical thinking is increasing. The case study above led to a novel kind of such support and TP technology suggests further research and development.

Extension of machinery for user-guidance is generalised as follows:

1. A few additional lines of code provide automatic detection of an error-pattern wherever a respective error occurs. Such additions are generally applicable: they refer to a particular set of theorems, and theorems are the means to justify any step of calculation in any topic of mathematics. *So the extended machinery generalises to all kinds of mathematics.*

2. An additional error-pattern, for instance dealing with cancelling fractions, carries over to any re-use of the respective theorems in any other application. For instance, error-patterns implemented for calculating with rational, carry over to *all* calculations with rational, because the respective program code serves both (due to

Lucas-Interpretation): (1) the creation of next-step-guidance and (2) the re-use in programs on more advanced problems. *So the machinery generalises over all re-uses in the course of systematic build-up of mathematics without additional code for user-guidance.*

3. Four dialogue rules extended user-guidance by three adaptive system reactions: (1) show a hint-page (and probably follow the links on this page), (2) suggest an incomplete next step (and check respective input) and (3) propose the rule to be applied. These four rules can easily be changed for other kinds of adaptive user-guidance; and, of course, the small number of rules indicates best prerequisites for well-structured extensions by many, many other rules. *So the dialogue machinery generalises and scales to high complexity in accordance to future requests.*

4. Additional interaction elements could improve the whole learning scenario: For example: step back to the last good point and show a tactic, make a list of examples (or list of pages) to solve for a specific error-pattern. Or transfer a problem automatically to a new worksheet (even from the list of proposed examples.)

Counters are the first point raising questions for further research and development. Counters occur in all of the rules in §4.7, see `EP_counter` and `help_counter_`. However, nothing has been said about these questions:

Which value does a counter start with? Always zero? If not zero, do counters take previous sessions into account? What is the correlation between `EP_counter` and `help_counter_`? If counters are stored for many sessions, what is relevant? Their average? Do these counters provide information useful for user-guidance and for assessment?

These questions and others cannot be bypassed on the way to a **user-model**.

Recording user-actions is already implemented in ISAC: The `EUIElements` represent steps which promote the construction of a solution within a logical context and a user not aware of the context will fail to make such a step', closely related to comprehensive mathematical activities.

So a **history** of user-actions is ready to be used for statistical analysis — but can the abundance of data be seriously related to cognitive processes involved in mathematics (Clancey [27], to problem-solving expertise? What structures and correlations can be expected from statistical analysis over large populations? What about comparison of different populations?

Hint-pages can pop up due to adaptive user-guidance like in Fig.4.2 on p.54 — but there is HTML-technology available with exciting new multimedia features (videos, pictures, diagrams, graphics...etc.). These shall connect individual thinking with support for multiple representations. This aspect is important because according to (Rittle-Johnson et al. [117]) multiple representation is key in the abstraction process. Further improvement for hint-pages could also be: highlighting patterns, taking actual example and solving parts of it, also as the following:

1. Getting more information from web/other sources
   (a) Show real world connections
   (b) Find a game, or other examples, forums, blogs..etc (by pointing at a word, or part of the formula)
2. Examples in the hint-page
   (a) Compare bad and good solutions, by counting the version of the learner till the end

(b) Count only a part of the whole example, where the learners get stuck (that is also possible in a new window)

(c) Count the example with concrete numbers

(d) Possibility to change numbers in a small (concrete) example (What happens if we put there one, two..etc)

(e) Show: when does the value change, when not.

(f) Pattern matching with colors. -highlighting errors, important points

Blended learning environments of a new kind can be assembled from components envisaged above. This opens new possibilities for complex learning activities: support for learning in a social environment; for involving the teacher (learners are influenced by the teachers external representation in both procedural and conceptual knowledge(Bills [11]), and also the individual way of thinking. People have individual strategies in solving mathematical problems and individual ways of learning (Rittle-Johnson et al. [117]).

Regarding error-patterns, and the interaction , interesting research questions arise (also from a cognitive science point of view).

1. Which errors occur together (is there maybe an underlying structure, or is it depending on the way of learning?), and how should it be assessed, that the learner can sometimes apply a rule, and sometimes not.

2. The current machinery in *ISAC* opens the possibility with small changes to design different learning scenarios and environments, to identify which errors do happen at identified factors (e.g.: To which extend does it depend from the environment, teacher, or the type of example? How can we support and make the process of learning abstraction easier? Why do specific errors happen?)

3. A general improvement for *ISAC* would be to identify which errors are due to miscalculation, or lack of concentration compared to "real" errors where the learner would need a wider explanation, not just a short hint.

Technology acceptance Technological environments change constantly and rapid at our times. Paradoxically to the fact that children and young adults have a higher and faster technology acceptance, in education the technology acceptance is slower than in other parts of life: i.e. daily in working life (See how many are using computers, smart-phones on a day to day basis, and how many computers are used in schools.) We still have to take into consideration, that future technological developments proceeds very fast in the direction of fusion of the following: human-centred and context sensitive design; going online, online platforms, social collaboration; different devices: computers, smart-phones, sensors, multi-touch, Apps, involving multimedia content (video, image, audio, text); personalised education in order to create an education that one can access on the basis of what she/he needs, at the time of need, and to learn at the speed that is comfortable for the individual. We believe, that an educational software, independent from its domain, should meet the challenge of this rapid development and flexibility. *ISAC* provides a possibility for a flexible technological environment to meet parts of the above mentioned criteria.

Pilot studies in regular classes have not been done yet, with good reasons: We do not expect significant results from a study limited to a few lessons. Rather, the strength of the error-pattern technology described in this thesis is embedding into complex learning scenarios — errors cannot be planned, rather errors should be handled at the moment they occur; the strength comes to bear with frequent use of this technology. For that reason the system would need to cover most of mathematics taught at high-school (which is not

the case in present ISAC). For instance, error-patterns for fractions are helpful again when exercising in different domains: e.g. integration in higher grades for some learners, but not for all: this is one of the strengths of the approach of ISAC. However, we did several case studies with colleagues, which led to one remarkable result: More frequently than expected users make several errors at once, and these combinations can not (yet?) be detected by the technology described. There are ideas, how to tackle this problem in a specific dialogue mode, where the learner is led into situations, where they cannot make more than one error; these ideas are out of scope of this thesis. The other question is the age of the learner: There is a need for younger learners, and also altering the systems for their age. As in most cases, technologies used in mathematical education are designed for older learners (Highfield and Goodwin [60]). This thesis presents the first implementation of error-patterns, and in order to conduct relevant field studies further implementations are necessary, and it needs long-term tests, how ISAC would adapt to different ages, because it is quite a fine graded detection of errors, how learners using ISAC perform compared to non-ISAC users. By analysing protocol of learners collected during learning sessions we can get new insight into learning about this domain.

Chapter 6

Summary and Conclusion

"I seldom end up where I wanted to go,
but almost always end up where I need to be."
Douglas Adams

6.1 Summary

Experience with the general R&D process showed that several iterations were necessary, in order to come from the starting question to an executable implementation in the ISAC prototype.

The question was as general as "What can cognitive science can contribute to the development of ISAC?". The iterated approach from this general question pays off by the fact, that ISAC is a prototype of an upcoming generation of educational mathematics assistants, which is based on Computer Theorem Proving.

As a prototype, ISAC required considerable effort for installation; and then interacting with the system was not as intuitive as promised. However, the essential features became apparent with some experience. Nevertheless, it was a challenge to cut down the expectations to what is realisable in a computer.

Personal experience with teaching mathematics in private lessons and in courses had sharpened personal expectations of how computer support in learning mathematics should look like.

So iterations proceeded by learning how to use limited technical means in order to accomplish a maximum of the expectations — and to advocate appropriate theories and respective references from Cognitive Science in order to provide scientific grounds for the expectations.

The contents of the thesis

1. Chapter introduced points common to cognitive science and to mathematics. The guiding questions were: what is mathematics, are there innate processes, what is the role of abstraction in learning, what is special about algebra and fraction, where do errors in mathematics originate from, and how we can use computers in mathematics education.

2. Chapter explained the existing features of the TP-based system: ISAC, its next step guidance. showing the starting state of its dialogue component.

3. Chapter investigated deeper the possible contributions from cognitive science: How do learners solve problems and calculate examples in the domain of fractions, what kind of knowledge is needed in mathematics, and what is needed for an effective

guiding/tutoring. The chapter also compared existing computer systems used in mathematical education. Summarizing the above described elements we established guidelines for ISAC's dialogue design. As having a focus on reacting on errors we also investigated the question whether errors are patterns like, and which error patterns could be useful for the dialogue in ISAC.

4. Chapter showed the concrete implementation of error-patterns, hint-pages, rewrite rules, fill-forms, dialogue rules and dialogue modes in ISAC.

5. Chapter compared real-world examples with the possibilities of error-detection in ISAC. The Chapter contained also how we can embed the above described findings into learning scenario, proposals for guided example, and as well as for future extending (e.g.: user models, pilot studies).

The implementation in ISAC was surprisingly little effort, after research in the cognitive science literature and cooperation with ISAC developers had led to a concise design in §4.2.

About hundred lines of code in SML define the error-pattern for addition of fractions and respective fill-forms; further error-patterns as shown in the appendix §B have not been implemented. Given these definitions according to the respective design, ISAC's mathematics engine detects error-patterns automatically; the code required for detection has been implemented by the ISAC development team.

Detection of error-patterns needs to be exploited by ISAC's dialogue module. This module is a rule-based engine from an expert system and dialogues are determined by rules, which can be fairly easily implemented. So it was tempting to start trials with these rules — however, we decided to confine this thesis to a minimum of rules, just as a demonstration of feasibility: Finally four rules were sufficient to determine the learning sequence with the hint-page and the fill-forms as described in §4.1.

However, only the hint-page can be demonstrated, not the fill-form (which would have required further R&D on ISAC's user interface).

Guidelines for further development are provided by this thesis: §4 give detailed descriptions which are meant as guidelines for successors in R&D, who want to implement further error patterns for other kinds of calculations in other areas of mathematics, and who want to extend the set of rules in ISAC's dialogue module.

In the course of further R&D details of the current design might change, so §4 has already by transferred into a wiki by the ISAC development team:
`http://www.ist.tugraz.at/isac/Guidelines_for_Dialog_Authoring`

6.2 Conclusion

The first attempt to amend ISAC' dialogues with expertise from cognitive science was successful: it has clarified foundations, and although not all the goals have been achieved, the advancements in technology are breaking new grounds.

What the thesis has clarified provides solid grounds for ISAC's cognitive design of dialogues, demonstrated fruitful in this thesis and giving direction to further R&D.

Concepts in cognitive science have been identified in learning mathematics, which can be modeled in TP-based software in a straight forward manner: this is demonstrated by "errors" and the implementation of "error patterns", further ideas mentioned in the thesis §5.3 are considered a glimpse on future possibilities.

On the other hand concepts in cognitive science have been identified, which are hard to model in software or even are principally not feasible in software: embodiment of learn-

ing, adapt to emotions, directly support the process of abstraction. These findings suggest particular care when designing complex user models in the future.

The terrain between the extremes, "straight forward implementation" and "principally not feasible in software", has been clarified within the limited scope of the thesis: reaction on *all* possible errors occurring during the learning process cannot be expected, because some errors are ad-hoc, some errors are hard to identify for technical reasons (for instance, if there are several conceptual errors, i.e. error patterns, in one input formula) or some kinds of errors have not yet identified. In particular, the relations between errors and respective patterns of occurrence, seem promising for research.

What has been not achieved results from several reasons.

The interaction on fill-forms is not implemented as described in §4.1. The reason for that is a technical feature required in ISAC's worksheet, which was detected missing during the work on the thesis, and which could not be provided in time.

The implemented error-patterns were not tested with students. §5.3 gave the reason "error cannot be planned in pilot studies"; or course, other reasons are still missing user models and more elaborated dialogues required for concisely proving advancements.

Many ideas have been mentioned but not realised, for instance adaptive and interactive hint pages. The reasons for that are simply lack of time; also the usability of ISAC's authoring tools is still very low. However, these considerations led to a more general design.

The advancements in technology are significant and demonstrate the potential of combining expertise from Cognitive Science with the novel features of Theorem-Proving technology. The error patterns for fractions

1. **generalise to virtually all other kinds of calculations**: to calculations with roots, with complex numbers, with trigonometric functions, to differentiation and integration, etc. Error patterns address a level of skills considered basic, but indispensable for succeeding in any formal discipline.

2. **transfer to all applications of calculations**: For instance, fractions occur in "real world problems" at high-school, in integration of polynomials, in equation solving, etc. And since systems like ISAC build all applications from elementary modules like simplifications of fractions, the mechanisms automatically carry over to any application in engineering and technology[1].

3. **Efficiency of the technology** is demonstrated by the fact, that only four additional rules suffice to expand the dialog module for handling error-patterns. The small number promises scalability of adaptive user guidance into a new level of complexity, significantly beyond the present state-of-the-art; §5.3 presents some ideas.

Advantages for research in Cognitive Science resulting from the advancements in TP-based technology have been identified in this thesis as follows.

1. **Cognitive design is free to focus structures of human learning** in joint development of educational math software: TP-based systems, in principle, model some essence of mathematics, which is reliable handling of logical relations between formal facts. Given that software structure, development is free to focus human aspects, e.g. of errors.

2. **Cognitive design has powerful services at disposal** when building upon TP-based technology: (1) automated check of (fairly free) user input (2) automated proposal of a next step when the learner gets stuck and (3) human readable format of all

[1] Observations from academic courses show, that there are students who would take profit from re-instruction on elementary skills, precisely at the occasion when they run into problems.

underlying mathematics knowledge. The design of error-patterns only gives a tiny preview to future possibilities.

3. **A new level of complexity of dialogues** comes within reach when exploiting the above mentioned services: Four (4!) additional rules were shown sufficient to exploit error detection (see "generalisation" and "transfer" above). The thesis identifies further exploitations: reflect on errors, give information *if requested*, refresh the knowledge at the moment where forgotten, etc.

4. **Novel opportunities for research in cognitive science** arise from tutoring systems with complex dialogues. With current technological and theoretical background there is little support for understanding and finding out the thoughts of the learner. ISAC, for instance, protocols high-lever interaction between system and learner, which result in a wealth of significant data.

Acknowledgements

I would like to express my very great appreciation to Dr. Walther Neuper from University of Graz, for his advices given in my questions regarding mathematics, education, and this thesis, and he has been a great help in understanding ISAC, and I am particularly grateful for his assistance. I would like to thank for Jürgen Markus Donko for helping with the German translation, and for all persons involved in the organisation of the MEi:CogSci program and curricula which enriched my view on research and science.

Bibliography

[1] S. Ainsworth. The functions of multiple representations. *Computers & Education*, 33:131–152, 1999.

[2] L. Amador. *Drools developers cookbook: over 40 recipes for creating a robust business rules implementation by using JBoss Drools rules*. Birmingham, U.K. : Packt, 2012.

[3] L. Amador. *Drools developers cookbook: over 40 recipes for creating a robust business rules implementation by using JBoss Drools rules*. Birmingham, U.K. : Packt, 2012.

[4] J.R. Anderson. Intelligent tutoring and high school mathematics. Technical Report 20, Carnegie Mellon University, Department of Psychology, 2008. http://repository.cmu.edu/psychology/20.

[5] C. Attard and M. Northcote. Teaching with technology. *Australian Primary Mathematics Classroom ISSN 1326-0286*, 17(1):29–32, 2012.

[6] R.-J. Back. Structured derivations: a unified proof style for teaching mathematics. *Formal Aspects of Computing*, 22(5):629–661, 2010.

[7] N. Balacheff. Future perspectives for research in the psychology of mathematics education. In *Mathematics and Cognition, A Research Synthesis by the International Group for the Psychology of Mathematics Education*, 1990.

[8] S. Barzilai and A. Zohar. How does information technology shape thinking? *Thinking Skills and Creativity*, 1:130–145, 2006.

[9] M. J. Beeson. Mathpert: Computer support for learning algebra, trig and calculus. In A. Voronkov, editor, *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR'92)*, volume 624 of *LNAI*, pages 454–456, St. Petersburg, Russia, July 1992. Springer Verlag. ISBN 3-540-55727-X.

[10] T. Ben-Zeev. The nature and origin of rational ediys in arithmetic thinking: Induction from examples and prior knowledge. *Cognitive Science*, 19:341–376, 1995.

[11] C. Bills. The influence of teachers representation on pupils mental representation. *Research in Mathematics Education*, 2, Issue 1:45–60, 2000.

[12] C. Blair, S. Gamson, S. Thorne, and D. Baker. Rising mean iq: Cognitive demand of mathematics education for young children, population exposure to formal schooling, and the neurobiology of the prefrontal cortex. *Intelligence*, 33:93–106, 2005.

[13] J. Bobis. Visualisation and the development of number sense with kindergarten children. In J. Mulligan and M. Mitchelmore, editors, *Children's Number Learning*, page 1733. Adelaide: AAMT & MERGA, 1996.

[14] P. Boero, T. Dreyfus, K. Gravemeijer, E. Gray, R. Hershkowitz, B. Schwarz, A. Sierpinska, and D. Tall. Abstraction: Theories about the emergence of knowledge structures. *Proceedings of the 26th annual conference of the International Group for the Psychology of Mathematics Education*, Vol. 1:113–138, 2002.

[15] G. Booker. *Theories of Mathematical Learning*, chapter 22 Constructing Mathematical Conventions Formed by the Abstraction and Generalization of Earlier Ideas: THe development of Initial Fraction Ideas, pages 241–266. Lawrence Erlbaum Associates, Inc., 1996.

[16] A. V. Borovik. *Mathematics under the microscope. Notes on cognitive aspects of mathematical practice*. Providence, RI: American Mathematical Society (AMS), 2010.

[17] R. M. Bottino and G. Chiappini. Advanced technology and learning environments. their relationships within the arithmetic problem-solving domain. In *Handbook of international research in mathematics education*. English, Lyn D. et al, 2002.

[18] T. Breiteig and B. Grevholm. The transition from arithmetic to algebra: to reason, explain, argue, generalize and justify. *Proceedings 30 th Conference of the International Group for the Psychology of Mathematics Education*, 2:225–232, 2006.

[19] J. S. Brown and R. R. Burton. Diagnostic models for procedural bugs in basic mathematical skills*. *Cognitive Science*, 2:155–192, 1978.

[20] J. S. Brown and K. VanLehn. Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science*, 4:379–426, 1980.

[21] N.G. Bruijn. Automath archive, 2012. URL `http://www.win.tue.nl/automath/`.

[22] G. E. Buckingham. Diagnostic and remedial teaching in first year algebra. *Chicago, IL: Northwestern Universiity School Education. Series*, 11, 1933.

[23] J.F. Cantlon, E.M. Brannon, E.J. Carter, and K. A. Pelphrey. Functional imaging of numerical processing in adults and 4-y-old children. *PLoS Biology*, 4, Issue 5.: 844–855, May 2006.

[24] D. W. Carraher. *Theories of Mathematical Learning*, chapter 14 Learning About Fraction, pages 241–266. Lawrence Erlbaum Associates, Inc., 1996.

[25] P. C. H. Cheng. Unlocking conceptual learning in mathematics and science with effective representational systems. *Computers in Education*, 33(2-3):109–130., 1999.

[26] M. T. H. Chi, M. Royb, and R. G. M. Hausmann. Observing tutorial dialogues collaboratively: Insights about human tutoring effectiveness from vicarious learning. *Cognitive Science*, 32:301–341, 2008.

[27] B. Clancey. Modeling the perceptual component of conceptual learning–a coordination perspective. In P. Grdenfors and P. Johansson, editors, *Cognition, Education and Communication Technology*, pages 109–146. Mahwah, NJ: Lawrence Erlbaum Associates, 2005.

[28] J. Clement. Observed methods for generating analogies in scientific problem solving. *Cognitive Science*, 12:563–586, 1988.

[29] P. Cobb, G. A. Goldin, and B. Greed. *Theories of Mathematical Learning*. Lawrence Erlbaum Associates, Inc., 1996.

[30] A. Collins. Cognitive apprenticeship. In Keith R. Sawyer, editor, *Cambridge Handbook of the Learning Sciences*. MIT Press, Washington University, St Louis, 2006. ISBN:9780521845540.

[31] K. Dahl and S. Nordqvist. *Zahlen, Spiralen und magische Quadrate*. Verlag Friedrich Oetinger, Hamburg, 2009.

[32] Arany Daniel. Matematikai verseny. URL `http://matek.fazekas.hu/portal/feladatbank/egyeb/feladatok/aranydani.html`.

[33] T. Danzig. *Number, the Language of Science*. New York: Free Press, 1954.

[34] G. Daroczy. Cognitive aspects of designing rule based dialogue guidance in educational mathematical assistant. In *MEi:CogSci Conferences, MEi:CogSci Conference 2012, Bratislava*, 2012. http://www.univie.ac.at/meicogsci/php/ocs/index.php/meicog/meicog2012/paper/view/367.

[35] G. Daroczy and W. Neuper. Error-patterns within next-step-guidance in tp-based educational systems. *eJMT the Electronic Journal of Mathematics and Technology*, 7(2), (Special Issue February):175–194, 2013.

[36] R.B Davis, E. Jockusch, and C McKnight. Cognitive processes in learning algebra. *The Journal of Childrens Mathematical Behaviour*, 2(1):10–320, 1987.

[37] E. De Corte. Technology-supported learning in a stage of transition: the case of mathematics. In *Lessons from Learning*, pages 113–123, 1993.

[38] M. Delazer, A. Ischebeck, F. Domahs, L. Zamarian, F. Koppelstaetter, C.M. Siedentopf, L. Kaufmann, T. Benke, and S. Felber. Learning by strategies and learning by drill-evidence from an fmri study. *Neuroimage*, 25:838–849, 2005.

[39] Coq development team. Coq 8.3 reference manual. http://coq.inria.fr/reman, 2010. INRIA.

[40] K. Devlin. *The Math Gene*. Basic Books, 2000.

[41] Oxford Dictionaries. "mathematics"., April 2010. URL `http://oxforddictionaries.com/definition/english/mathematics`.

[42] Oxford Dictionaries. "dialogue"., April 2010. URL `http://oxforddictionaries.com/definition/english/dialogue`.

[43] Oxford Dictionaries. "fraction". 25 march 2013, April 2010. URL `<http://oxforddictionaries.com/definition/english/fraction>`.

[44] T. Dreyfus. Advanced mathematical thinking. In *Mathematics and Cognition, A Research Synthesis by the International Group for the Psychology of Mathematics Education*, 1990.

[45] L. D. Edwards. Embodying mathematics and science: microworlds as representations. *J. Math. Behav.*, 17(1):53–78, 1998.

[46] R.M Falcon. Integration of a cas/dgs as a cad system in the mathematics curriculum for architecture students. *Int. J. Math. Educ. Sci. Technol.*, 42(6):737–750, 2011.

[47] L. Fazio and R. Siegler. Teaching fractions. *Educational Practices Series, International Academy of Education (IAE)*, 22, 2011.

[48] J. Fleener. *Mathematics and Knowledge*. University of North Carolina at Chapel Hill, 1980. URL `http://books.google.at/books?id=gRUPOAAACAAJ`.

[49] R. Gelman and C. Gallistel. *The Child's Understanding of Number.* Cambridge, MA: Harvard University Press., 1978.

[50] M. Giaquinto. *Visual Thinking in Mathematics*. Oxford University Press, 2007.

[51] J. D. Godino. Mathematical concepts, their meaning and understanding. *Proceedings of XX Conference of the International Group for the Psychology of Mathematics Education*, 2:417–425, 1996.

[52] M. Goldgruber. Algebraische Simplifikation mittels Rewriting in ISAC. Master's thesis, University of Technology, Institute for Softwaretechnology, Graz, Austria, Sept 2003.
http://www.ist.tugraz.at/projects/isac/publ/DA-M02-main.ps.gz.

[53] M. Gordon, R. Milner, and C. P. Wadsworth. *Edinburgh LCF: A Mechanised Logic of Computation*, volume 78 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

[54] R.H. Graber, D. Ansari, G. Reishofer, E. Stern, F. Ebner, and C. Neuper. Individual differences in mathematical compentence predict paretal brain activation during mental calculation. *Neuroimage*, 38:346–356, 2007.

[55] R. H. Grabner, D. Ansari, G. Reishofer, E. Stern, F. Ebner, and C. Neuper. Individual differences in mathematical competence predict parietal brain activation during mental calculation. *NeuroImage*, 38:346–356, 2007.

[56] R. H. Grabner, A. Ischebeck, G. Reishofer, K. Koschutnig, M. Delazer, F. Ebner, and C. Neuper. Fact learning in complex arithmetic and figural-spatial tasks: The role of the angular gyrus and its relation to mathematical competence. *Human Brain Mapping*, 30(9):2936–2952, 2009. ISSN 1097-0193. doi: 10.1002/hbm.20720. URL `http://dx.doi.org/10.1002/hbm.20720`.

[57] A. Griesmayer. Architecture and Knowledge-Represenation of the Web-based Math-Learning-System ISAC. Master's thesis, University of Technology, Institute for Softwaretechnology, Graz, Austria, Oct 2003.
http://www.ist.tugraz.at/projects/isac/publ/da-griesmayer.pdf.

[58] J. Hadamard. *The Mathematician's Mind: The Psychology of Invention in the Mathematical Field*. Princeton University Press, 1996.

[59] R. Hembree. The nature, effects, and relief of mathematics anxiety. *Journal for Research in Mathematics Education*, 21(1):33–46, 1990.

[60] K. Highfield and K. Goodwin. A review of recent research in early mathematics learning and technology. In K. Makar M. Goos, R. Brown, editor, *Navigating Currents and Charting Directions (Proceedings of the 31st Annual Conference of the Mathematics Education Research Group of Australasia)*, pages 259 – 264. MERGA, 2008.

[61] N. Idris and L. M. Narayanan. Error patterns in addition and substraction of fractions among form two students. *Journal of Mathematics Education*, 4(2):35–54, 2011.

[62] M. Jennison and K. Beswick. Student attitude, student understanding and mathematics anxiety, shaping the future of mathematics education. In *proceedings of the 33rd annual conference of the Mathematics Education Research Group of Australasia, 3 - 7 July 2010, Fremantle, Western Australia, pp. 280-288.*, 2010.

[63] S. Karnel. Grösste gemeinsame Teiler in Polynomringen und Implementierung im $\mathcal{ISAC}$-Projekt. Master's thesis, University of Technology, Institute of Mathematics, Graz, Austria, August 2002.
http://www.ist.tugraz.at/projects/isac/publ/GGTs-von-Polynomen.ps.gz.

[64] M. Kienleitner. Towards "nextstep userguidance" in a mechanized math assistant. Master's thesis, IICM, Graz University of Technology, 2012. http://www.ist.tugraz.at/projects/isac/publ/mkienl_bakk.pdf.

[65] M. Kienleitner and F. Kober. *Logging of High-Level Steps in a Mechanized Math Assistant*. PhD thesis, Bakkalaureate Thesis at IICM, Graz University of Technology, 2012.

[66] C. Kieran. Cognitive processes involved in learning school algebra. In *Mathematics and Cognition, A Research Synthesis by the International Group for the Psychology of Mathematics Education*, 1990.

[67] K. M. Kim and W. Kang. An analysis on the repeated error patterns in division of fraction by elementary students. *J. Korea Soc. Math. Educ. Ser.*, 11(1):1–19, 2008.

[68] F. Kober. Logging high-level interactions in a mechanized math assistant. Master's thesis, IICM, Graz University of Technology, 2012. http://www.ist.tugraz.at/projects/isac/publ/fkober_bakk.pdf.

[69] K. R. Koedinger and B. A. Maclaren. Developing a pedagogical domain theory of early algebra problem solving. Technical report, Carnegie Mellon University, 2002.

[70] K. R. Koedinger, M. W. Alibabi, and M. L. Nathan. Trade-offs between grounded and abstract representations: Evidence from algebra problem solving. *Cognitive Science*, 32:366–397, 2008.

[71] A. Koestler. *The Act of Creation*. Hutschinson, 1964.

[72] A: Kok. Understanding the technology enhanced learning environments from a cognitive perspective. *International Education Studies*, 2(4), 2009.

[73] A. Krempler. Architectural design for integrating an interactive dialogguide into a mathematical tutoring system. Master's thesis, University of Technology, Institute for Softwaretechnology, Graz, Austria, March 2005.
http://www.ist.tugraz.at/projects/isac/publ/da-krempler.pdf.

[74] A. Krempler and W. Neuper. Formative assessment for user guidance in single stepping systems. In Michael E. Aucher, editor, *Interactive Computer Aided Learning, Proceedings of ICL08*, Villach, Austria, 2008.
http://www.ist.tugraz.at/projects/isac/publ/icl08.pdf.

[75] K. Kucian, T. Loenneker, T. Dietrich, M. Dosch, E. Martin, and M. Aster. Impaired neural networks for approximate calculation in dyscalculic children: a functional mri study. *Behavioural and Brain Function*, 2:3:1, 2006.

[76] C. Laborde, C. Kynigos, K. Hollebrands, and R. Strasser. Teaching and learning geometry with techonology. In A. Gutierrez & Pl. Boero, editor, *Handbook of research on the psychology of mathematics education: Past, present and future*, pages 275–304. Rotterdam, The Netherlands: Sense Publischer, 2006.

[77] S. P. Lajoie. Computer environments as cognitive tools for enhancing learning. In S. P. Lajoie, editor, *Computers as Cognitive Tools*. Hillsdale, NJ: Lawrence Erlbaum Associates., 1993.

[78] I. Lakatos. Proofs and refutations: the logic of mathematical discovery. *British Journal for the Philosophy of Science*, 14:1–25, 1963. Edited by John Worrall and Elie Zahar, Cambridge University Press, 1976.

[79] G. Lakoff and R.E: Nunez. *Where Mathematics Comes From*. Basic Books: ISBN-10: 0465037712, 2001.

[80] O. Linnebo. Platonism in the philosophy of mathematics. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2011 edition, 2011.

[81] L. Ma. *Knowing and teaching elementary mathematics: Teachers understanding of fundamental mathematics in China and the United States*. Mahwah, NJ: Lawrence Erlbaum Associates., 1999.

[82] S. Mathan, J. R. Anderson, A.T. Corbett, and C. Lovett. Recasting the feedback debate: Benefits of tutoring error detection and correction skills. In *In*, pages 13–20. Press, 2003.

[83] N. M. McNeil and M. W. Alibali. Youll see what you mean: Students encode equations based on their knowledge of arithmetic. *Cognitive Science*, 28:451–466, 2004.

[84] D. Meade. *Getting Started with Maple, 3rd ed.* Wiley, 2009.

[85] E. Melis and J. Siekmann. An intelligent tutoring system for mathematics. In L. Rutkowski, J. Siekmann, R. Tadeusiewicz, and L.A. Zadeh, editors, *Seventh International Conference Artificial Intelligence and Soft Computing (ICAISC)*, number 3070, in LNAI, page 91101. Springer-Verlag, 2004. doi: 10.1007/978-3-540-24844-6_12.

[86] E. R. Michene. Understanding understanding mathematics. *Cognitive Science*, 2: 361–383, 1978.

[87] Robin Milner, Mads Tofte, Robert Harper, and David MacQueen. *The Definition of Standard ML (Revised)*. The MIT Press, Cambridge, London, 1997.

[88] M. C. Mitchelmore and P. White. Abstraction in mathematics: Conflict, resolution and application. *Mathematics Education Research Journal*, 7(1):50–68, 1995.

[89] M. C. Mitchelmore and P. White. Teaching mathematics concepts: Instruction for abstraction. *ICME-10 Proceedings*, 2008.

[90] A. Mora, E. Marida, and R. Eixarch. Random learning units using wiris quizzes in moodle. *Int. J. Math. Educ. Sci. Technol.*, 42(6):751–763, 2011.

[91] J. Moses. Macsyma:a personal history. Invited Presentation in Milestones in Computer Algebra, Tobago, 2008. URL `http://esd.mit.edu/Faculty_Pages/moses/Macsyma.pdf`.

[92] W. Neuper. Angewandte Mathematik und Fachtheorie. Technical Report 357, IMST – Innovationen Machen Schulen Top!, University of Klagenfurt, Institute of Instructional and School Development (IUS), 9010 Klagenfurt, Sterneckstrasse 15, 2006. http://imst.uni-klu.ac.at/imst-wiki/index.php/Angewandte_Mathematik_und_Fachtheorie.

[93] W. Neuper. Angewandte Mathematik und Fachtheorie. Technical Report 683, IMST – Innovationen Machen Schulen Top!, University of Klagenfurt, Institute of Instructional and School Development (IUS), 9010 Klagenfurt, Sterneckstrasse 15, 2007. http://imst.uni-klu.ac.at/imst-wiki/index.php/Angewandte_Mathematik_und_Fachtheorie_2006/2007.

[94] W. Neuper. Common grounds for modelling mathematics in educational software. *Int. Journal for Technology in Mathematics Education*, 17(3), 2010.

[95] W. Neuper. Automated generation of user guidance by combining computation and deduction. In Pedro Quaresma and Ralph-Johan Back, editors, Proceedings First Workshop on *CTP Components for Educational Software,* Wrocław, Poland, 31th July 2011, volume 79 of *Electronic Proceedings in Theoretical Computer Science*, pages 82–101. Open Publishing Association, 2012. doi: 10.4204/EPTCS.79.5.

[96] W. Neuper. On the emergence of tp-based educational math assistants. volume 7, pages 110–129, February 2013. URL `https://php.radford.edu/~ejmt/ContentIndex.php#v7n2`. Special Issue "TP-based Systems and Education".

[97] W. Neuper and J. Reitinger. Begreifen und Mechanisieren beim Algebra Einstieg. Technical Report 1063, IMST – Innovationen Machen Schulen Top!, University of Klagenfurt, Institute of Instructional and School Development (IUS), 9010 Klagenfurt, Sterneckstrasse 15, 2008.
http://imst.uni-klu.ac.at/imst-wiki/index.php/Begreifen_und_Mechanisieren_beim_Algebra-Einstieg.

[98] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[99] R. Nunez. Embodied cognition as grounding for situatedness and context in mathematics education. *Educational Studies in Mathematics*, 39 (1-3):45–66, 1999.

[100] R. Nunez. Conceptual metaphors and the cognitive foundation of mathematics. In *B.Baaqui & P. Pang (Eds.) Metaphor and the Cognitive Foundation of Mathematics: Actual Infinity and Human Cognition*, 2003.

[101] R. Nunez. Embodied cognition and the nature of mathematics: Language, gesture, and abstraction. In *Proceeding of the 26th Annual Conference of the Cognitive Science Society (pp.36-37)*, 2004.

[102] R. Nunez. What is mathematics? pauli, jung, and contemporary cognitive science. In *H. Altmanspacher & H. Primas (Eds.), Recasting Reality: Wolfgang Paulis Philosophical Ideas and Contemporary Science, New York: Springer*, 2008.

[103] R. Nunez. No innate number line in the human bain. *Journal of Cross-Cultural Psychology,*, 45(4):651–668, 2011.

[104] National Council of Teachers of Mathematics and National Research Council Mathematical Sciences Education Board. *The Nature and Role of Algebra in the K-14 Curriculum:Proceedings of a National Symposium*. The National Academies Press, 1998. ISBN 9780309061476. URL `http://www.nap.edu/openbook.php?record_id=6286`.

[105] R. J. Payne and H.R. Squibb. Algebra mal-rules and cognitive accounts of error. *Cognitive Science*, 14:445–481, 1990.

[106] R. Prank, M. Issakova, D. Lepp, E. Tonisson, and V. Vaiksaar. T-algebra - interactive learning environment for expression manipulation. In *7th International Conference on Technology in Mathematics Teaching*, volume 1, pages 26–29, Bristol, UK, July 2005.

[107] Quotes. Quotes on mathematics (collection), . URL `http://www.quotationspage.com/subjects/mathematics/`.

[108] Quotes. Quotes on mathematics (collection), . URL `http://www.brainyquote.com/quotes/keywords/mathematics.html`.

[109] Quotes. Quotes on mathematics (collection), . URL `http://www.math.okstate.edu/~wli/teach/fmq.html`.

[110] Quotes. Quotes on mathematics (collection), . URL http://www.m4maths.com/maths-quotes.php.

[111] Quotes. Quotes on mathematics (collection), . URL http://www.goodreads.com/quotes/tag/mathematics.

[112] Quotes. Quotes on mathematics (collection), . URL http://www.searchquotes.com/quotation.

[113] F. Reif. Interpretation of scientific or mathematical concepts: Cognitive issues and instructional implication. *Cognitive Science*, 11:395–416, 1987.

[114] A. Renkl. Learning from worked-out examples: A study on individual differences. *Cognitive*, 21:1–29, 1997.

[115] S. Ritter, J.R. Anderson, K.R. Koedinger, and A. Corbett. Cognitive tutor: Applied reserach in mathematics education. *Psychonomic Bulletin & Review*, 14 (2):249–255, 2007.

[116] B. Rittle-Johnson and R. S. Siegler. The relation between conceptual and procedural knowledge in learning mathematics: A review. In *The development of mathematical skills. Studies in developmental psychology*. England: Psychology Press/Taylor & Francis (UK), 1998.

[117] B. Rittle-Johnson, R.S. Siegler, and M.W. Alibali. Developing conceptual understanding and procedural skill in mathematics: An iterative process. *Developing conceptual understanding and procedural skill in mathematics: An iterative process.*, 93:346–362, 2001.

[118] J. Ročnik. Trials with tp-based programming for interactive course material. volume 7, pages 91–109, February 2013. URL https://php.radford.edu/~ejmt/ContentIndex.php#v7n2. Special Issue "TP-based Systems and Education".

[119] B. Rogoff, B. Matusov, and S. White. Models of teaching and learning: Participation in a community of learners. In In D. Olson & N. Torrance, editor, *The Handbook of Cognition and Human Development*, pages 388–414. Oxford, UK, Blackwell, 1996.

[120] R. K. Skemp. *The Psychology of Learning Mathematics*. Harmondsworth, England: Penguin, 1986.

[121] D. Sleeman. An attempt to understand students understanding of basic algebra*. *Cognitive Science*, 8:387–412, 1984.

[122] D. Sleeman, A.E. Kelly, R. Martinak, R.D. Ward, and J.L. Moore. Studies of diagnosis remediution and with high school algeb students. *Cognitive Science*, 13: 551–568, 1989.

[123] A. Sloman. If learning maths requires a teacher, where did the first teacher come from? In *Paper for Mathematical Cognitive Symposium, AISB2010, March 29-30*, 2010.

[124] D. Strom, V. Kemeny, R. Lehrer, and E. Forman. Visualizing the emergent structure of childrens mathematical argument. *Cognitive Science*, 25:733–773, 2001.

[125] M.B. Swan. Dealing with misconceptions in mathematics. *Issues in Mathematics Teaching*, pages 147–165, 2001.

[126] J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12:257–285, 1988.

[127] M. Taylor and D. L. Schwartz. Physically distributed learning: Adapting and reinterpreting physical environments in the development of fraction concepts. *Cognitive Science*, 29(4):587–625, 2005. ISSN 1551-6709. doi: 10.1207/s15516709cog0000_15. URL http://dx.doi.org/10.1207/s15516709cog0000_15.

[128] D. Tirosh, Even R., and Robinson M. Simplifying algebraic expressions: Teacher awareness and teaching approaches,. *Educational Studies in Mathematics*, 35:51–64, 1998.

[129] J. C. Turner, Midgley C., Meyer D. K., Gheen M., Anderman E. M., and Kang Y. T. he classroom environment and students' reports of avoidance strategies in mathematics: A multimethod study. *Journal of Educational Psychology,*, 94 (1):88–106, 2002.

[130] K. VanLehn. Analogy eventy: How examples are used during problem solving. *Cognitive Science*, 22:347–388, 1998.

[131] K. VanLehn, A.C. Graesser, G.T. Jackson, P. Jordan, A. Olney, and C. P. Rose. When are tutorial dialogues more effective than reading? *Cognitive Science*, 31: 3–62, 2007.

[132] A.W. Vincent, M.M. Aleven, and K.R. Koedinger. An effective metacognitive strategy: learning by doing and explaining with a computer-based cognitive tutor. *Cognitive Science*, 26:147–179, 2002.

[133] R. Vogel. Patterns - a fundamental idea of mathematical thinking and learning. *ZDM, Zentralbl. Didakt.*, 37(5):445–449, 2005.

[134] E. Walker. Mutual peer tutoring: A collaborative addition to the cognitive tutor algebra i. accepted as a young researcher's track paper at. In *the International Conference on Artificial Intelligence and Education*, 2005.

[135] E. Warren and T. Cooper. Using repeating patterns to explore functional thinking: Elizabeth warren and tom cooper lead us through a series of teaching activities desing to develop young childrens algebraic thinking. *Australian Primary Mathematics Classroom ISSN 1326-0286*, 2012.

[136] E. Wenger. *Artificial intelligence and tutoring systems: Computational and cognitive approaches to the communication of knowledge*. Los Altos, CA: Morgan Kaufmann Publishers., 1987.

[137] J. J. Williams and T. Lombrozo. The role of explanation in discovery and generalization: Evidence from category learning. *Cognitive Science*, 34:776–806, 2010.

[138] S. Wolfram. *The Mathematica Book*. Wolfram Research Inc., 1999.

[139] E. Yetkin. Students difficulties in learning elementary mathematics. *http://www.tpdweb.umi.com/tpweb*, 2003.

[140] R. M. T. Young and OShea. Errors in childrens subtraction. *Cognitive Science*, 5(2): 153177, 1981.

[141] J. Zhang. The nature of external representations problem in solving. *Cognitive Science*, 21:179–217, 1997.

Appendices

Appendix A

General Collection on Examples in the Domain of Fractions

The appendix contains a detailed representation of the data implemented (or to be implemented) in this thesis for use by ISAC's dialogue guide. The representation serves as an interface to mathematics authors.

Factorisation

$\frac{3+6 \cdot a}{3+9 \cdot b} =$

$\frac{16+4 \cdot a}{4a} =$

$\frac{x \cdot y+y}{z \cdot y+y} =$

$\frac{a-b}{b-a} =$

Multiplication of Fraction

$\frac{x^2+6x+9}{x^2+6x+8} \cdot \frac{3x+6}{3x+9} =$

$(x^2 - 9) \cdot \frac{3}{4x-12} =$

$\frac{3a}{5} \cdot \frac{a^2}{21} =$

$\frac{x^2-4}{9x^6} \cdot \frac{6x^3}{2x-4} =$

Binomial Form

$\frac{25 \cdot x^2+40 \cdot x-9}{-45 \cdot x^2+4 \cdot x+1} =$

$\frac{10 \cdot x^2+68 \cdot x-14}{x^2+4 \cdot x-21} =$

$\frac{32 \cdot x^2+48 \cdot x+16}{-32 \cdot x^2+12 \cdot x+20} =$

$\frac{-35 \cdot x^2+66 \cdot x-16}{21 \cdot x^2+43 \cdot x-14} =$

Addition and Subtraction of Fraction

$\frac{a}{5} + \frac{b}{5} =$

$\frac{8}{x^4} + \frac{16}{x^4} =$

$\frac{3}{7 \cdot a} + \frac{2}{14 \cdot a} =$

$\frac{2 \cdot x}{y^2 \cdot z^2} + \frac{3}{y \cdot z} =$

Dividing Fraction

$\frac{\frac{2 \cdot x - 6 \cdot y}{3}}{\frac{3 \cdot x^2 - 9 \cdot x \cdot y}{2 \cdot y}} =$

$\frac{a + \frac{1}{2}}{b - \frac{1}{2}} =$

$\frac{\frac{4 \cdot a \cdot b}{c}}{\frac{2 \cdot a \cdot b}{c}} =$

$\frac{5 \cdot (x - y)^2}{\frac{3 \cdot x - 3 \cdot y}{a}} =$

Simplification of Fraction

$\frac{4}{2x + 4} =$

$\frac{3x^3 + x^2 + 3x + 1}{15x^2 + 5x} =$

$\frac{1}{2x + 3y} + \frac{3}{8x + 12y} - \frac{1}{6x + 9y} =$

$\frac{1}{x - 1} + \frac{2x^2 - 3}{x^2 - 1} - \frac{2x}{x + 1} =$

Mixed examples

$\frac{1}{x + 5} + 19 + \frac{1}{x - 2} - \frac{2x + 3}{x^2 + 3x - 10} =$

$\frac{3x + 1}{x + 1} + \frac{x + 2}{x - 1} - \frac{x + 5}{x^2 - 5} =$

$\frac{4x}{4x - 6y} + \frac{2x}{6x + 9y} - \frac{3x}{24x^2 - 54y^2} =$

$\left(\frac{6x - 3}{x - y} - \frac{6x + 3}{x + y}\right) : \frac{36x^2 - 16}{x + y} =$

Appendix B

Error-Patterns, Fill-Patterns and Fill-Forms, Hint-Pages

B.1 Error-Patterns

Definition error-pattern [35]: Given a triple (id, P, R) with *id* a string called **identifier**, P a set of terms with equality called **patterns** and R a set of **rewrite-rule**s, this triple is called an **error-pattern** iff

(i) $\forall p \in P.\ \exists r \in R.\ match\,(lhs(p), lhs(r)) \neq \emptyset$. This r is called the rule $\mathbf{belonging\_to\ p}$

(ii) $\forall r \in R.\ \exists p \in P.\ match\,(lhs(p), lhs(r)) \neq \emptyset$

Although implemented at different locations in the ML code, respective error-patterns and fill-patterns are listed together in order to support design considerations. For the same reason all the patterns are first given by LaTeXformulas, while the code is represented "verbatim" in order to ease copying to the ML code.

1. Addition of fractions (Same can be used for substraction):

   (a) $\frac{a}{b} + \frac{c}{d} = \frac{a+c}{b+d}$,

   (b) $a + \frac{b}{c} = \frac{a+b}{c}$,

   (c) $\frac{a+b}{c+d} = \frac{a}{c} + \frac{b}{d}$,

```
val errpats =
  [("addition-of-fractions",
     [parse_patt thy "(?a / ?b + ?c / ?d) = (?a + ?c)/(?b +?d)",
      parse_patt thy "(?a / ?b + ?c / ?d) = (?c + ?a)/(?b +?d)",
      parse_patt thy "(?a / ?b + ?c / ?d) = (?c + ?a)/(?d +?b)",
      parse_patt thy "(?a / ?b + ?c / ?d) = (?a + ?c)/(?d +?b)",
      ( (b) $a+\frac{b}{c}=\frac{a+b}{c}$)
      parse_patt thy "(?a + ?c)/(?d +?b)= (?a / ?d)+(?c / ?b)",
      parse_patt thy "(?a + ?c)/(?d +?b)= (?a / ?b)+(?c / ?d)",
      parse_patt thy "(?a + ?c)/(?d +?b)= (?c / ?d)+(?a / ?b)",
      parse_patt thy "(?a + ?c)/(?d +?b)= (?c / ?b)+(?a / ?d)",
      parse_patt thy "?a + (?b /?c)= (?a + ?b)/ ?c"
      ],
     [@{thm rat_add}, @{thm rat_add_assoc}, @{thm rat_add1},
       @{thm rat_add1_assoc}, @{thm rat_add2}, @{thm rat_add2_assoc},
       @{thm rat_add3},  @{thm rat_add3_assoc}])
  ]: errpat list;
```

Theorems for Addition:

```
ML {*
```

```
  @{thm rat_add};
  @{thm rat_add_assoc};
  @{thm rat_add1}; (*this is the only rule for addition
                   in case of variables*)
  @{thm rat_add1_assoc};
  @{thm rat_add2};
  @{thm rat_add2_assoc};
  @{thm rat_add3};
  @{thm rat_add3_assoc};
*}
val it = "?a is_const ==> ?b is_const ==> ?c is_const
==> ?d is_const ==>
  ?a / ?c + ?b / ?d = (?a * ?d + ?b * ?c) / (?c * ?d)": thm
val it = "?a is_const ==> ?b is_const ==> ?c is_const
==> ?d is_const ==>
  ?a / ?c + (?b / ?d + ?e)
  = (?a * ?d + ?b * ?c) / (?d * ?c) + ?e": thm
?a / ?c + ?b / ?c = (?a + ?b) / ?c": thm
val it = "?a is_const ==> ?b is_const ==> ?c is_const ==>
  ?a / ?c + (?b / ?c + ?e) = (?a + ?b) / ?c + ?e": thm
val it = "?a is_const ==> ?b is_const ==> ?c is_const ==>
  ?a / ?c + ?b = (?a + ?b * ?c) / ?c": thm
val it = "?a is_const ==> ?b is_const ==> ?c is_const ==>
  ?a / ?c + (?b + ?e) = (?a + ?b * ?c) / ?c + ?e": thm
val it = "?a is_const ==> ?b is_const ==> ?c is_const ==>
  ?a + ?b / ?c = (?a * ?c + ?b) / ?c": thm
val it = "?a is_const ==> ?b is_const ==> ?c is_const ==>
  ?a + (?b / ?c + ?e) = (?a * ?c + ?b) / ?c + ?e": thm
```

2. Multiplication and division of fractions:

   (a) $\frac{a}{c} \cdot \frac{b}{c} = \frac{a \cdot b}{c}$,

   (b) $\frac{a}{b} \cdot c = \frac{a \cdot c}{b}$,

   (c) $\frac{c}{a-b} \cdot \frac{d}{b} = \frac{c \cdot d}{a-2b}$

   (d) $\frac{a}{b} \cdot \frac{c}{b} = \frac{a+c}{b}$

```
val errpats =
  [("multiplication-of-fractions",
    [parse_patt thy "(?a / ?c) * (?b / ?c) = (?a * ?b) / ?c",
     parse_patt thy "(?a / ?b) * ?c = (?a * ?c) / (?b * ?c)",
     parse_patt thy "(?a / ?c) * (?b / ?c) = (?b * ?a) / ?c",
     parse_patt thy "(?a / ?b) * (?c / ?b) = (?a + ?c) / ?b",
     parse_patt thy "(?c / ( ?a - ?b) ) * (?d / ?b)
     = (?c * ?d) / ( ?a - 2 * ?b)"],
     [@{thm rat_mult}, @{thm rat_mult2} (*...*) ])]: errpat list;
```

Theorems for multiplication:

```
ML {*
  @{thm rat_mult};
  @{thm rat_mult2};
  (*...*)
*}
val it = "?a / ?b * (?c / ?d) = ?a * ?c / (?b * ?d)": thm
val it = "?a / ?b * ?c = ?a * ?c / ?b": thm
```

3. Division:
   $\frac{a}{b} \cdot \frac{c}{b} = \frac{a+c}{b}$

```
val errpats =
  [("division",
     [parse_patt thy "(?a / ?b) / (?c / ?d)= (?a / ?c) / (?b / ?d)",
     parse_patt thy "(?a / ?b) / (?c / ?d)= (?a * ?c) / (?b * ?d)",
 ],
     [@{thm ...}, @{thm ...}])
   ("addition"), [(*patterns*)], [(*theorems*)]]: errpat list;
```

4. Cancellation of fractions:

   (a) $\frac{a+b}{a} = b$
   (b) $\frac{a+b}{b} = a$
   (c) $\frac{a+b}{a+c} = \frac{b}{c}$
   (d) $\frac{a+c}{b+c} = \frac{a}{b}$
   (e) $\frac{a+b}{b+c} = \frac{a}{c}$
   (f) $\frac{a+b}{c+a} = \frac{b}{c}$
   (g) $\frac{a \cdot b+c}{a} = b + c$
   (h) $\frac{a \cdot b+c}{b} = a + c$
   (i) $\frac{a+b \cdot c}{b} = a + c$,
   (j) $\frac{a+b \cdot c}{c} = a + b$,
   (k) $\frac{a+bx}{c+dx} = \frac{a+b}{c+d}$
   (l) $\frac{a}{c} + \frac{a}{c} = \frac{a+b}{2 \cdot c}$
   (m) $\frac{a}{c} + \frac{a}{c} + \frac{d}{c} = \frac{a+b+d}{3 \cdot c}$
   (n) $\frac{a}{b} + \frac{c}{2 \cdot b} = \frac{a+c}{3 \cdot b}$

```
val errpats =
  [("cancel",
     [parse_patt thy "(?a + ?b)/?a = ?b",
      parse_patt thy "(?a + ?b)/?b = ?a",
      parse_patt thy "(?a + ?b)/(?b + ?c) = ?a / ?c",
      parse_patt thy "(?a + ?c)/(?b + ?c) = ?a / ?b",
      parse_patt thy "(?a + ?c)/(?b + ?c) = ?a / ?c",
      parse_patt thy "(?a + ?b)/(?c + ?a) = ?b / ?c",
      parse_patt thy "(?a*?b + ?c)/?a = ?b + ?c",
      parse_patt thy "(?a*?b + ?c)/?b = ?a + ?c",
      parse_patt thy "(?a + ?b*?c)/?b = ?a + ?c",
      parse_patt thy "(?a + ?b*?c)/?b = ?a + ?b",
      parse_patt thy "(?a + ?b*?e)/(?c + ?d*?e) = (?a + ?b)/(?c + ?d)"],
     [@{thm real_times_divide_1_eq},  @{thm real_times_divide_1_eq}])
   ("addition"), [(*patterns*)], [(*theorems*)]]: errpat list;
```

Theorems for Cancellation:

```
ML {*
  @{thm frac_eq_eq};
  @{thm divide_cancel_left};
  @{thm divide_cancel_right};
*}
val it = "?y \= (0?'a) ==> ?z \= (0?'a)
==> (?x / ?y = ?w / ?z) = (?x * ?z = ?w * ?y)": thm
val it = "(?c / ?a = ?c / ?b) = (?c = (0?'a)  ?a = ?b)": thm
val it = "(?a / ?c = ?b / ?c) = (?c = (0?'a)  ?a = ?b)": thm
```

5. Special Cases with 0:

   (a) $c + c \cdot b = c \cdot (0 + b)$
   (b) $\frac{b}{c} - \frac{a}{c} = \frac{b-a}{0}$
   (c) $\frac{a}{0}$

```
val errpats =
  [("special_0", ()
     [parse_patt thy " ?c+ ( ?c * ?b ) = ?c * ?b ",
```

```
      parse_patt thy "(?b / ?c - ?a / ?c = ?b - ?a",
      parse_patt thy "?a / ?b  = ?a / 0",
      ]
     [@{thm name_special_0}, @{thm name_special_0}])
   ("special_0"), [(*patterns*)], [(*theorems*)]]: errpat list;
```

6. Special Cases with 1:
   (a) $\frac{a-b}{b-a} = 1$
   (b) $-\frac{a}{b} = \frac{a}{b}$
   (c) $\frac{a}{b-x} = \frac{-a}{b-x} = -1 \cdot \frac{a}{b-x}$
   (d) $\frac{a}{a} = 0$
   (e) $\frac{1}{1} = 0$
   (f) $-a + b = -(a + b)$
   (g) $-a + b = a + b$
   (h) $a - b \cdot c = (a - b) \cdot c$
   (i) $a + b \cdot c = (a + b) \cdot c$

```
val errpats =
  [("special_1", (**)
     [parse_patt thy "(?a - ?b)/( ?b - ?a) = 1",
      parse_patt thy "-(?a / ?b) = ( ?a / ?b)",
      parse_patt thy "-?a + ?b = - ?a - ?b",
      parse_patt thy "-?a + ?b = ?a + ?b",
      parse_patt thy "(?a / ( ?b - ?c ) = - ?a / ( ?b - ?c )",
      parse_patt thy "(?a / ( ?b - ?c ) = -1 * ( ?a / ( ?b - ?c )) ",
      parse_patt thy "?a - ( ?b * ?c )= ( ?a - ?b )* c",
      parse_patt thy "?a + ( ?b * ?c )= ( ?a + ?b )* c",
      parse_patt thy "(?a / ?a) = 0 "],
     [@{thm special_one},  @{thm special_1}])
   ("special_one"), [(*patterns*)], [(*theorems*)]]: errpat list;
```

7. Binomial Form:
   (a) $\frac{a-b}{a^2-b^2} = a - b$
   (b) $(\frac{a}{b} + c^2 = \frac{a^2}{b^2} + c^2$
   (c) $(a + b)^2 = a^2 + b^2$
   (d) $(a - b)^2 = a^2 + b^2$
   (e) $(a - b)^2 = a^2 - b^2$

```
val errpats =
  [("binom", (**)
     [parse_patt thy "(?a - ?b)/ (?a ^{2} - ?b^{2}) = ?a - ?b",
     parse_patt thy "(?a + ?b)^{2} = ?a^{2} + ?b^{2}",
     parse_patt thy "(?a - ?b)^{2} = ?a^{2} - ?b^{2}",
     parse_patt thy "(?a - ?b)^{2} = ?a^{2} + ?b^{2}",
     parse_patt thy "(?a / ?b)+ ?c ^{2} =  ?a^{2} / ?b^{2} + ?c^{2} "
     ],
     [@{thm binom_1}, @{thm binom_2}])
   ("binom"), [(*patterns*)], [(*theorems*)]]: errpat list;
```

8. Changing between forms:
   (a) $a\frac{b}{c} = \frac{a+b}{c}$
   (b) $a\frac{b}{c} = \frac{a \cdot d}{c \cdot d}$

(c) $a\frac{b}{c} = \frac{a \cdot d + c}{c \cdot d}$

```
val errpats =
  [("changeform", (**)
     [parse_patt thy "a(?b / ?c)/?a = ( ?a + ?b ) / ?c",
      parse_patt thy "a(?b / ?c)/?a = ( ?a * ?b ) / ( ?c * ?b )",
      parse_patt thy "a(?b / ?c)/?a = ( ?a * ?b + ?c ) / ( ?c * ?b )"],
     [@{thm changeform_1}, @{thm changeform_2}])
   ("changeform"), [(*patterns*)], [(*theorems*)]]: errpat list;
```

B.2 Fill-Patterns and Fill-Forms

Definition [35]: Fill-Pattern Given a rewrite-rule r and an ordered set (enumerable by $\mathcal{N}$, the natural numbers) of triples $\phi = \{\phi_i.\ 1 \leq i \leq n \in \mathcal{N} \ \wedge \ \phi_i = (id_i, p_i, \mathcal{P}_i(id_\epsilon))\}$, where id_i is a string called **identifier**, p_i a term with equality called **fill-in-pattern** and $\mathcal{P}_i(id_\epsilon)$ a set of error-patterns' identifiers, such a set is called a **fill-pattern of r** if

(i) $\forall i.\ 1 \leq i \leq n \Rightarrow lhs(p_i) = lhs(r)$

(ii) $\forall i.\ 1 \leq i \leq n \Rightarrow match(rhs(p_i), rhs(r)) \neq \emptyset$ and the $lhs(p_i)$ "contain less placeholders with growing i"

For terms f and f_I we say '$\mathbf{f_I}$ **is_filled_into** ϕ ' iff $f \rightarrow_r f_I$.

1. Fill Pattern: Addition of fractions:

(a) $\frac{a}{b} + \frac{c}{d} = \frac{a+c}{b+d}$,
$\frac{a}{b} + \frac{c}{d} = \frac{\dddot{\cdot}}{\dddot{\cdot}} + \frac{\dddot{\cdot}}{\dddot{\cdot}}$,
$\frac{a}{b} + \frac{c}{d} = \frac{\dddot{a}}{\dddot{b}} + \frac{\dddot{c}}{\dddot{d}}$,
$\frac{a}{b} + \frac{c}{d} = \frac{\dddot{\cdot}\cdot d}{\dddot{\cdot}\cdot d} + \frac{\dddot{\cdot}\cdot b}{\dddot{\cdot}\cdot b}$,
$\frac{a}{b} + \frac{c}{d} = \frac{\dddot{\cdot}\cdot d}{\dddot{b}} + \frac{\dddot{\cdot}\cdot b}{\dddot{c}}$,
$\frac{a}{b} + \frac{c}{d} = \frac{a \cdot d}{\dddot{b}} + \frac{\dddot{\cdot}\cdot b}{\dddot{c}}$,
$\frac{a}{b} + \frac{c}{d} = \frac{\dddot{\cdot}+\dddot{\cdot}}{\dddot{\cdot}}$,
$\frac{a}{b} + \frac{c}{d} = \frac{a+c}{\dddot{d}}$,
$\frac{a}{b} + \frac{c}{d} = \frac{\dddot{\cdot}\cdot d + \dddot{\cdot}\cdot b}{\dddot{b}}$,
$\frac{a}{b} + \frac{c}{d} = \frac{a + c \cdot b}{b \cdot d}$,

(b) $a + \frac{b}{c} = \frac{a+b}{c}$,
$a + \frac{b}{c} = \frac{\ddot{\cdot}}{\ddot{\cdot}} + \frac{\ddot{\cdot}}{\ddot{\cdot}}$,
$a + \frac{b}{c} = \frac{\ddot{\cdot}}{\ddot{\cdot}} + \frac{b}{c}$,
$a + \frac{b}{c} = \frac{\ddot{\cdot}}{c} + \frac{b}{c}$,
$a + \frac{b}{c} = \frac{\ddot{\cdot}\cdot c}{c} + \frac{b}{c}$,
$a + \frac{b}{c} = \frac{\ddot{\cdot}+\ddot{\cdot}}{c}$,
$a + \frac{b}{c} = \frac{\ddot{\cdot}\cdot c + \ddot{\cdot}}{c}$,
$a + \frac{b}{c} = \frac{a \cdot c + b}{c}$,

(c) $\frac{a}{c} + \frac{b}{c} = \frac{a+b}{2 \cdot c}$
$\frac{a}{c} + \frac{b}{c} = \frac{\ddot{\cdot}+\ddot{\cdot}}{\ddot{\cdot}}$
$\frac{a}{c} + \frac{b}{c} = \frac{\ddot{\cdot}+\ddot{\cdot}}{c}$
$\frac{a}{c} + \frac{b}{c} = \frac{a+b}{\ddot{\cdot}}$
$\frac{a}{c} + \frac{b}{c} = \frac{a+\ddot{\cdot}}{c}$

(d) $\frac{a}{c} + \frac{b}{c} + \frac{d}{c} = \frac{a+b+d}{3 \cdot c}$
$\frac{a}{c} + \frac{b}{c} = \frac{\ddot{\cdot}+\ddot{\cdot}+\ddot{\cdot}}{\ddot{\cdot}}$
$\frac{a}{c} + \frac{b}{c} = \frac{a+\ddot{\cdot}+\ddot{\cdot}}{c}$
$\frac{a}{c} + \frac{b}{c} = \frac{a+b+\ddot{\cdot}}{\ddot{\cdot}}$

(e) $\frac{a}{b} + \frac{c}{d \cdot b} = \frac{a+c}{(d+1) \cdot b}$
$\frac{a}{b} + \frac{c}{d \cdot b} = \frac{\ldots}{d \cdot b} + \frac{c}{d \cdot b}$
$\frac{a}{b} + \frac{c}{d \cdot b} = \frac{a \cdot \ldots + \ldots}{\ldots \ldots}$
$\frac{a}{b} + \frac{c}{d \cdot b} = \frac{\ldots \cdot d + \ldots}{d \cdot b}$
$\frac{a}{b} + \frac{c}{d \cdot b} = \frac{\ldots \cdot d + c}{d \cdot \ldots}$

```
ML {*
val fillpat =
  ([("fill-addition-first1",
       parse_patt @{theory Rational}
       "(?a / ?b + ?c / ?d)
       = ( _ * ?d + _ *?b ) / ( _ * _ )",
         [" name1", " name2", " name3" ]),
     ("fill-addition-first2",
       parse_patt @{theory Rational}
       "(?a / ?b + ?c / ?d)
       = (?a * ?d + ?c *?b ) / (?b * ?d) ",
 [" name1", " name2", " name3" ]),

     ("fill-addition-first3",
       parse_patt @{theory Rational}
       "(?a / ?b + ?c / ?d)
       = (?a * ?d / ?b *?d ) + (?c * ?b / ?b * ?d )",
         [" name1", " name2", " name3" ]),


     ("fill-addition-first4",
       parse_patt @{theory Rational}
       "(?a / ?b + ?c / ?d) =  (?a * ?d + ?c *?b ) / (?b * ?d)",
         [" name1", " name2", " name3" ]),
     ("fill-addition-first5",
       parse_patt @{theory Rational}
       "(?a / ?b + ?c / ?d) = (?a * ?d + ?c *?b ) / (?b * ?d)",
         [" name1", " name2", " name3" ]),

     ("fill-addition-second1",
       parse_patt @{theory Rational}
       "?a + (?b /?c) = ( _ * _ + _ )/ ?c",
        [" name1", " name2", " name3" ]),
     ("fill-addition-second2",
       parse_patt @{theory Rational}
       "?a + (?b /?c) = ( _ * _ )/ _ + ( _ / ?c)",
        [" name1", " name2", " name3" ]),
     ("fill-addition-second3",
       parse_patt @{theory Rational}
       "?a + (?b /?c) = ( ?a * _ + _ )/ ?c",
        [" name1", " name2", " name3" ]),
     ("fill-addition-second4",
       parse_patt @{theory Rational}
       "?a + (?b /?c) = ( _* _ )/ ?c + ( ?b / ?c)",
        [" name1", " name2", " name3" ]),
     ("fill-addition-second5",
       parse_patt @{theory Rational}
       "?a + (?b /?c) = ( ?a *?c +_ )/ ?c",
        [" name1", " name2", " name3" ]),
     ("fill-addition-second6",
       parse_patt @{theory Rational}
       "?a + (?b /?c) = (?a * _ )/ ?c + ( ?b / ?c)",
        [" name1", " name2", " name3" ]),


   NONE): fillpat;
*}
```

2. Pattern for multiplication:

   (a) $\frac{a}{c} \cdot \frac{b}{c} = \frac{a \cdot b}{c}$, possible answers:

$\frac{a}{c} \cdot \frac{b}{c} = \frac{\ldots}{\ldots}$
$\frac{a}{c} \cdot \frac{b}{c} = \frac{a \cdot b}{\ldots}$
$\frac{a}{c} \cdot \frac{b}{c} = \frac{\ldots}{c \cdot c}$

(b) $a \cdot \frac{b}{c} = \frac{a \cdot b}{a \cdot c}$, possible answers:
$a \cdot \frac{b}{c} = \frac{\ldots}{\ldots}$
$a \cdot \frac{b}{c} = \frac{\ldots}{c}$
$a \cdot \frac{b}{c} = \frac{a \cdot \ldots}{c}$

(c) $\frac{c}{a-b} \cdot \frac{d}{b} = \frac{c \cdot d}{a-2b}$ possible answers:
Here it can have one of the same answers as by (a)

(d) $\frac{a}{b} \cdot \frac{c}{b} = \frac{a+c}{b}$ possible answers:
Here it can have one of the same answers as by (a)

```
ML {*
val fillpat =
  ([("fill-multiplication-first1",
       parse_patt @{theory Rational}
       "(?a / ?c) * (?b / ?c) = (_ * _) / (_ * _)",
         [" name1", " name2", " name3" ]),
    ("fill-multiplication-first2",
       parse_patt @{theory Rational}
       "(?a / ?c) * (?b / ?c) = (?a * _) / ( ?c * _)",
         [" name1", " name2", " name3" ]),
    ("fill-multiplication-first3",
       parse_patt @{theory Rational}
       "(?a / ?c) * (?b / ?c) = ( ?a * ? b) / ( ?c * _)",
         [" name1", " name2", " name3" ]),
    ("fill-multiplication-first4",
       parse_patt @{theory Rational}
       "(?a / ?c) * (?b / ?c) = (?a * _) / ( ?c * ?c)",
         [" name1", " name2", " name3" ]),

    ("fill-multiplication-second1",
       parse_patt @{theory Rational}
       "(?a / ?b) * ?c = (_ * _) / _ ",
         [" name1", " name2", " name3" ]),
    ("fill-multiplication-second1",
       parse_patt @{theory Rational}
       "(?a / ?b) * ?c = ( _ * _ ) / ?b ",
         [" name1", " name2", " name3" ]),
    ("fill-multiplication-second1",
       parse_patt @{theory Rational}
       "(?a / ?b) * ?c = ( _ * ?c) / ?b ",
         [" name1", " name2", " name3" ]),

    ("fill-multiplication-third1",
       parse_patt @{theory Rational}
       "(?a / ?c) * (?b / ?d) = ( _ * _ ) / ( _ * _ )",
        [" name1", " name2", " name3" ]),
    ("fill-multiplication-third1",
       parse_patt @{theory Rational}
       "(?a / ?c) * (?b / ?d) = ( _ * _ ) / ( ?c * ?d )",
        [" name1", " name2", " name3" ],
    ("fill-multiplication-third1",
       parse_patt @{theory Rational}
       "(?a / ?c) * (?b / ?d) = ( _ * ?b ) / ( ?c * _ )",
        [" name1", " name2", " name3" ]),


    ("fill-multiplication-fourth1",
       parse_patt @{theory Rational}
       "(?c / ( ?a - ?b) ) * (?d / ?b)
       = ( ?c * ?d )/ (?a * ?b - ?b * ?b)",
         [" name1", " name2", " name3" ]),
```

```
    ("fill-multiplication-fourth1",
       parse_patt @{theory Rational}
       "(?c / ( ?a - ?b) ) * (?d / ?b)
       = ( ?c * ?d )/ ( _ * ?b - _ * ?b)",
         [" name1", " name2", " name3" ]),
    ("fill-multiplication-fourth1",
       parse_patt @{theory Rational}
       "(?c / ( ?a - ?b) ) * (?d / ?b)
       = ( _ * _ )/ (?a * ?b - ?b * ?b)",
         [" name1", " name2", " name3" ]),


   NONE): fillpat;
*}
```

- $\frac{a}{c} \cdot \frac{b}{c} = \frac{a \cdot b}{c}$

```
val errpat =
  ("frac-mult-same-denom",
    [parse_patt thy "(?a / ?c) * (?b / ?c) = (?a * ?b) / ?c"],
    [@{thm times_divide_times_eq}
    (* "?x / ?y * (?z / ?w) = ?x * ?z / (?y * ?w)" *),
     @{thm rat_mult} (* "?a / ?b * (?c / ?d) =
     ?a * ?c / (?b * ?d)" *)]): errpat;
```

Fill-patterns for *times_divide_times_eq* (the same as for rewrite-rule *rat_mult*):

- $\frac{a}{c} \cdot \frac{b}{c} = \frac{a \cdot b}{\ldots} \quad | \quad = \frac{a \cdot \ldots}{c \cdot \ldots} \quad | \quad = \frac{a \cdot b}{c \cdot c}$

```
val fillpats =
  ([ ("fill-frac-mult-same-denom-1",
      parse_patt @{theory Rational}
      "(?a / ?b) * (?c * ?d) = (?a * ?c) / _",
      "frac-mult-same-denom"),
     ("fill-frac-mult-same-denom-2",
      parse_patt @{theory Rational}
      "(?a / ?b) * (?c * ?d) = (?a * _) / (?b * _)",
      "frac-mult-same-denom"),
     ("fill-frac-mult-same-denom-3",
      parse_patt @{theory Rational}
      "(?a / ?b) * (?c * ?d) = (?a * ?c) / (?b * ?d)",
      "frac-mult-same-denom")],
   SOME hint-page): fillpat list;
```

3. Patterns for division:

   (a) $\frac{a}{b} : \frac{c}{d} = \frac{a:c}{b:c}$ possible answers:

   (b) $\frac{a}{b} : \frac{c}{d} = \frac{\ldots}{\ldots} \cdot \frac{\ldots}{\ldots}$

   (c) $\frac{a}{b} : \frac{c}{d} = \frac{a}{b} \cdot \frac{\ldots}{\ldots}$

   (d) $\frac{a}{b} : \frac{c}{d} = \frac{a}{\ldots} \cdot \frac{d}{\ldots}$

   (e) $\frac{a}{b} : \frac{c}{d} = \frac{a}{\ldots} \cdot \frac{d}{c}$

```
ML {*
val fillpat =
  ([("fill-division-left-num",
       parse_patt @{theory Rational}
       "(?a / ?b) / (?c / ?d) = (?a * _ ) / ( _ / ?c)",
         [" name1", " name2", " name3" ]),
    ("fill-division-left-den",
       parse_patt @{theory Rational}
       "(?a / ?b) / (?c / ?d) = ( _ * _ ) / (?b / ?c)",
         [" name1", " name2", " name3" ]),
```

```
    ("fill-division-none",
       parse_patt @{theory Rational}
       "(?a / ?b) / (?c / ?d) = (?a * _ ) / (?b / ?c)",
        [" name1", " name2", " name3" ]),
   NONE): fillpat;
*}
```

4. Cancellation of fractions:

   (a) $\frac{a+b}{a} = b$,
   $\frac{a+b}{a} = \frac{..}{a} + ..$,
   $\frac{a+b}{a} = \frac{b}{..} + 1$

   (b) $\frac{a+b}{b} = a$,
   $\frac{a+b}{a} = \frac{a}{.} + 1$
   $\frac{a+b}{a} = \frac{.}{b} + 1$

   (c) $\frac{a+b}{a+c} = \frac{b}{c}$,

   (d) $\frac{a+c}{b+c} = \frac{a}{b}$,

   (e) $\frac{a+b}{b+c} = \frac{a}{c}$,

   (f) $\frac{a+b}{c+a} = \frac{b}{c}$,

   (g) $\frac{a+bx}{c+dx} = \frac{a+b}{c+d}$
   $\frac{a+b}{c+a} = \frac{a+b}{..+..}$
   $\frac{a+b}{c+a} = \frac{..+..}{c+.a}$
   $\frac{a+b}{c+a} = \frac{a+..}{c+.a}$

   (h) $\frac{a.b+c}{a} = b + c$,

   (i) $\frac{a.b+c}{b} = a + c$,
   $\frac{a.b+c}{a} = \frac{..}{..} + ..$,
   $\frac{a.b+c}{a} = \frac{..}{a} + 1$,
   $\frac{a.b+c}{a} = \frac{b}{a} + ..$,

   (j) $\frac{a+b.c}{b} = a + c$,

   (k) $\frac{a+b.c}{c} = a + b$,
   $\frac{a+b.c}{a} = .. + \frac{..}{..}$,
   $\frac{a+b.c}{a} = 1 + \frac{..}{a}$,
   $\frac{a+b.c}{a} = .. + \frac{b}{a}$,

   (l) $\frac{a+b}{c+d} = \frac{a}{c} + \frac{b}{d}$
   $\frac{a+b}{c+d} = \frac{..+..}{..+..}$
   $\frac{a+b}{c+d} = \frac{a+..}{c+..}$
   $\frac{a+b}{c+d} = \frac{a+b}{c+..}$

```
ML {*
val fillpat =
  ([("fill-cancel-left-add1",
  parse_patt @{theory Rational}
  "(?a * (?b + ?c)) / (?a * (?d + ?c))
  = (_ + _) / (_ + _)", "cancel"),
    ("fill-cancel-left-add2",
    parse_patt @{theory Rational}
    "(?a * (?b + ?c)) / (?a * (?d + ?c))
    = (?b + _) / (_ + _)", "cancel"),
    ("fill-cancel-left-add3",
    parse_patt @{theory Rational}
    "(?a * (?b + ?c)) / (?a * (?d + ?c))
```

```
    = (?b + _) / (?d + _)", "cancel"),
    ("fill-cancel-left-add4",
    parse_patt @{theory Rational}

    "(?a * (?b + ?c)) / (?a * (?d + ?c))
     = (?b + ?c) / (?d + ?c)", "cancel")
  ("fill-cancel-left-num",
      parse_patt @{theory Rational} "(?a * ?b) / (?a * ?c)
       = _ / ?c",
        ["cancel", "cancel+/+", "cancel0/_"]),
    ("fill-cancel-left-den",
      parse_patt @{theory Rational} "(?a * ?b) / (?a * ?c)
       = ?b / _",
        ["cancel", "cancel+/+", "cancel0/_"]),
    ("fill-cancel-none",
      parse_patt @{theory Rational} "(?a * ?b) / (?a * ?c)
      = ?b / ?c",
        ["cancel", "cancel+/+", "cancel0/_"])],
   NONE): fillpat;
   SOME "exp_Etc_Frac_Hint_EP5"): fillpat;
*}
```

5. Special Cases with 0:

   (a) $\frac{b}{c} - \frac{a}{c} = \frac{b-a}{0}$
   $\frac{b}{c} - \frac{a}{c} = \frac{..-..}{..}$
   $\frac{b}{c} - \frac{a}{c} = \frac{..-..}{c}$
   $\frac{b}{c} - \frac{a}{c} = \frac{a-..}{c}$

   (b) $\frac{a}{0}$

   (c) $c + c \cdot b = c \cdot (0 + b)$
   $c + c \cdot b = .. \cdot (.. + \frac{..}{..})$
   $c + c \cdot b = .. \cdot (1 + \frac{..}{..})$
   $c + c \cdot b = c \cdot (.. + \frac{..}{b})$
   $c + c \cdot b = c \cdot (1 + \frac{..}{b})$

```
   ML {*
   val fillpat =
     ([("fill-special-null-first-1",
          parse_patt @{theory Rational}
          " ?c+ ( ?c * ?b ) = _ * ( _ + _ )",
            [" name1", " name2", " name3" ]),
       ("fill-special-null-first-2",
          parse_patt @{theory Rational}
          " ?c+ ( ?c * ?b ) = _ * ( 1 + _ )",
            [" name1", " name2", " name3" ]),
       ("fill-special-null-first-3",
          parse_patt @{theory Rational}
          " ?c+ ( ?c * ?b ) = ?c * ( 1 + _ )",
            [" name1", " name2", " name3" ]),

        ("fill-special-null-second-1",
          parse_patt @{theory Rational}
          "(?b / ?c - ?a / ?c = ( _ - _ )/ _ ",
            [" name1", " name2", " name3" ]),
        ("fill-special-null-second-1",
          parse_patt @{theory Rational}
          "(?b / ?c - ?a / ?c = ( _ - _ )/ ?c",
            [" name1", " name2", " name3" ]),
        ("fill-special-null-second-1",
          parse_patt @{theory Rational}
          "(?b / ?c - ?a / ?c = ( ?b - _ )/ ?c",
            [" name1", " name2", " name3" ]),

        ("fill-special-null-third-1",
```

```
        parse_patt @{theory Rational}
        "?a / ?b = _ / _ ",
         [" name1", " name2", " name3" ]),
      ("fill-special-null-third-1",
        parse_patt @{theory Rational}
        "?a / ?b = ?a / _ ",
         [" name1", " name2", " name3" ]),
      ("fill-special-null-third-1",
        parse_patt @{theory Rational}
        "?a / ?b = _ / ?b ",
         [" name1", " name2", " name3" ]),

    NONE): fillpat;
*}
```

6. Special Cases with 1 or -1:

   (a) $\frac{a-b}{b-a} = 1$
   $\frac{a-b}{b-a} = \frac{-1\cdot(..+..}{b-a}$
   $\frac{a-b}{b-a} = \frac{a-b}{(-1)\cdot(..+..)}$
   $\frac{a-b}{b-a} = \frac{-1\cdot(-a+..}{b-a}$
   $\frac{a-b}{b-a} = \frac{a-b}{(-1)\cdot(-b+..)}$

   (b) $-\frac{a}{b} = \frac{a}{b}$
   $-\frac{a}{b} = .. \cdot \frac{\cdot\cdot}{\cdot\cdot}$
   $-\frac{a}{b} = \left(-\frac{1}{\cdot\cdot}\right) \cdot \frac{a}{\cdot\cdot}$
   $-\frac{a}{b} = \frac{(-1)\cdot a}{\cdot\cdot}$

   (c) $\frac{a}{b-x} = \frac{-a}{b-x} = -1 \cdot \frac{a}{b-x}$
   $\frac{a}{b-x} = \frac{\cdot\cdot}{(-1)\cdot(..+..)}$
   $\frac{a}{b-x} = \frac{\cdot\cdot}{(-1)\cdot(-b+..)}$
   $\frac{a}{b-x} = \frac{\cdot\cdot}{(-1)\cdot(..-..)}$
   $\frac{a}{b-x} = \frac{\cdot\cdot}{(-1)\cdot(x-..)}$
   $\frac{a}{b-x} = \frac{a}{(..)\cdot(x-b)}$

   (d) $\frac{a}{a} = 0$
   $\frac{a}{a} = ..$
   $\frac{a}{a} = 1$
   $\frac{a}{a} = .. \cdot \frac{\cdot\cdot}{\cdot\cdot}$
   $\frac{a}{a} = a \cdot \frac{1}{\cdot\cdot}$
   $\frac{a}{a} = .. \cdot \frac{1}{a}$

   (e) $-a + b = -(a + b)$

   (f) $-a + b = a + b$
   $-a + b = .. \cdot (.. + ..)$
   $-a + b = -1 \cdot (a - ..)$
   $-a + b = 1 \cdot (-a + ..)$

```
ML {*
val fillpat =
  ([("fill-special-one-first-1",
       parse_patt @{theory Rational}
       "(?a - ?b)/( ?b - ?a) = (( _ - _ )/
       (-1* ( _ - _ ))",
         [" name1", " name2", " name3" ]),
         ("fill-special-one-first-2",
       parse_patt @{theory Rational}
       "(?a - ?b)/( ?b - ?a) = ((?a - _ )/
        (-1* ( - _ + _ ))",
         [" name1", " name2", " name3" ]),
         ("fill-special-one-first-3",
```

```
   parse_patt @{theory Rational}
   "(?a - ?b)/( ?b - ?a) = ((?a - ?b)/
    (-1* ( _ - _))",
     [" name1", " name2", " name3" ]),
     ("fill-special-one-first-4",
   parse_patt @{theory Rational}
   "(?a - ?b)/( ?b - ?a) = (( _ - ?b)/
   ( _ * ( -?b - ?a))",
     [" name1", " name2", " name3" ]),
     ("fill-special-one-first-5",
   parse_patt @{theory Rational}
   "(?a - ?b)/( ?b - ?a) = ((?a - ?b)/
    (-1* ( ?a - _ ))",
     [" name1", " name2", " name3" ]),


 ("fill-special-one-second-1",
   parse_patt @{theory Rational}
   "-(?a / ?b) = - 1* ( _ / _ )",
     [" name1", " name2", " name3" ]),
 ("fill-special-one-second-2",
   parse_patt @{theory Rational}
   "-(?a / ?b) = - _ / ?b",
     [" name1", " name2", " name3" ]),
 ("fill-special-one-second-3",
   parse_patt @{theory Rational}
   "-(?a / ?b) = - 1* ( _ / _ )",
     [" name1", " name2", " name3" ]),

 ("fill-special-one-third-1",
   parse_patt @{theory Rational}
   "(?a / ( ?b - ?c ) = - 1*  _ / ( _ - _)",
     [" name1", " name2", " name3" ]),
 ("fill-special-one-third-2",
   parse_patt @{theory Rational}
   "(?a / ( ?b - ?c ) = _ / (-1 ( _- _ )",
     [" name1", " name2", " name3" ]),
("fill-special-one-third-3",
   parse_patt @{theory Rational}
   "(?a / ( ?b - ?c ) = -1 *?a / ( _ - ?b)",
     [" name1", " name2", " name3" ]),
 ("fill-special-one-third-4",
   parse_patt @{theory Rational}
   "(?a / ( ?b - ?c ) = ?a / (-1 ( _ - ?c)",
     [" name1", " name2", " name3" ]),

 ("fill-special-one-fourth-1",
   parse_patt @{theory Rational}
   "?a - ( ?b * ?c ) = _ - _ * ?c",
     [" name1", " name2", " name3" ]),
 ("fill-special-one-fourth-2",
   parse_patt @{theory Rational}
   "?a - ( ?b * ?c ) = ?a - _ * ?c",
     [" name1", " name2", " name3" ]),

 ("fill-special-one-fifth-1",
   parse_patt @{theory Rational}
   "-?a + ?b  = = -1* (_  - _ )",
     [" name1", " name2", " name3" ]),
 ("fill-special-one-fifth-2",
   parse_patt @{theory Rational}
   "-?a + ?b  = = 1* (- _ + _ )",
     [" name1", " name2", " name3" ]),
 ("fill-special-one-fifth-3",
   parse_patt @{theory Rational}
   "-?a + ?b  = = -1* (?a - _)",
     [" name1", " name2", " name3" ]),
```

```
    ("fill-special-one-fifth-4",
       parse_patt @{theory Rational}
       "-?a + ?b  = = 1* (- _ + ?b)",
         [" name1", " name2", " name3" ]),

    ("fill-special-one-six-1",
       parse_patt @{theory Rational}
       "(?a / ?a) = 1* ( _ / _ )",
         [" name1", " name2", " name3" ]),
    ("fill-special-one-six-2",
       parse_patt @{theory Rational}
       "(?a / ?a) = _ * (?a / ?a )",
         [" name1", " name2", " name3" ]),
    ("fill-special-one-six-3",
       parse_patt @{theory Rational}
       "(?a / ?a) = 1* (?a / _ )",
         [" name1", " name2", " name3" ]),

   NONE): fillpat;
*}
```

7. Binomial Form:

(a) $\frac{a-b}{a^2-b^2} = a - b$
$\frac{a-b}{a^2-b^2} = \frac{..-..}{(..+..)\cdot(..-..)}$
$\frac{a-b}{a^2-b^2} = \frac{a-..}{(a+..)\cdot(a-..)}$
$\frac{a-b}{a^2-b^2} = \frac{a-b}{(a+b)\cdot(a-..)}$
$\frac{a-b}{a^2-b^2} = \frac{1}{..+..}$
$\frac{a-b}{a^2-b^2} = \frac{1}{a+..}$

(b) $(a+b)^2 = a^2 + b^2$
$(a+b)^2 = ..^2 + 2 \cdot .. \cdot .. + ..^2$
$(a+b)^2 = a^2 + 2 \cdot .. \cdot .. + b^2$
$(a+b)^2 = ..^2 + 2 \cdot a \cdot b + ..^2$

(c) $(a-b)^2 = a^2 - b^2$

(d) $(a-b)^2 = a^2 + b^2$
$(a+b)^2 = ..^2 - 2 \cdot .. \cdot .. + ..^2$
$(a+b)^2 = a^2 - 2 \cdot .. \cdot .. + b^2$
$(a+b)^2 = ..^2 - 2 \cdot a \cdot b + ..^2$

```
ML {*
val fillpat =
  ([("fill-binom-first-1",
       parse_patt @{theory Rational}
       "(?a - ?b)/ (?a ^{2} - ?b^{2})=
       (_ - _)/ (( _ - ?b)( _ + ?b))",
         [" name1", " name2", " name3" ]),
   ("fill-binom-first-2",
       parse_patt @{theory Rational}
       "(?a - ?b)/ (?a ^{2} - ?b^{2})=
       (_)/ ( _ + _ )",
         [" name1", " name2", " name3" ]),
   ("fill-binom-first-3",
       parse_patt @{theory Rational}
       "(?a - ?b)/ (?a ^{2} - ?b^{2})=
       (?a - _ )/ ((?a - _ )(?a + _ ))",
         [" name1", " name2", " name3" ]),
   ("fill-binom-first-4",
       parse_patt @{theory Rational}
       "(?a - ?b)/ (?a ^{2} - ?b^{2})=
       (_)/ ( _ + ?b)",
         [" name1", " name2", " name3" ]),
```

```
      ("fill-binom-first-5",
           parse_patt @{theory Rational}
           "(?a - ?b)/ (?a ^{2} - ?b^{2})=
           (?a - ?b )/ ((?a - ?b )(?a + _ ))",
             [" name1", " name2", " name3" ]),
       ("fill-binom-first-6",
           parse_patt @{theory Rational}
           "(?a - ?b)/ (?a ^{2} - ?b^{2})=
           (1)/ ( _ + ?b)",
             [" name1", " name2", " name3" ]),

        ("fill-binom-second-1",
           parse_patt @{theory Rational}
           "(?a + ?b)^{2} = _ ^{2} +2 * _ * _ + _^ {2}?,
             [" name1", " name2", " name3" ]),
             ("fill-binom-second-2",
           parse_patt @{theory Rational}
           "(?a + ?b)^{2} = _ ^{2} +2 * ?a * _ + _^{2}?,
             [" name1", " name2", " name3" ]),
             ("fill-binom-second-3",
           parse_patt @{theory Rational}
           "(?a + ?b)^{2} = ?a ^{_} + _ * ?a * ?b + ?b^{2}?,
             [" name1", " name2", " name3" ]),

         ("fill-binom-third-1",
           parse_patt @{theory Rational}
           "(?a - ?b)^{2} = _ ^{2} - 2 * _ * _ + _^ {2}?,
             [" name1", " name2", " name3" ]),
             ("fill-binom-third-2",
           parse_patt @{theory Rational}
           "(?a - ?b)^{2} = _ ^{2} - 2 * ?a * _ + _^{2}?,
             [" name1", " name2", " name3" ]),
             ("fill-binom-third-3",
           parse_patt @{theory Rational}
           "(?a - ?b)^{2} = ?a ^{_}  -  _ * ?a * ?b + ?b^{2}?,
             [" name1", " name2", " name3" ]),
       NONE): fillpat;
    *}
```

8. Changing between forms:
   (a) $a\frac{b}{c} = \frac{a+b}{c}$
   (b) $a\frac{b}{c} = \frac{a \cdot d}{c \cdot d}$
   (c) $a\frac{b}{c} = \frac{a \cdot d + c}{c \cdot d}$
   $a\frac{b}{c} = \frac{a \cdot .. + ..}{..}$
   $a\frac{b}{c} = \frac{.. \cdot c + ..}{c}$
   $a\frac{b}{c} = \frac{a \cdot c + ..}{..}$
   $a\frac{b}{c} = \frac{a \cdot c + . b}{..}$

```
ML {*
val fillpat =
  ([("fill-changeform-left-num",
       parse_patt @{theory Rational} "a(?b / ?c)/?a  = _ / ?c",
         [" name1", " name2", " name3" ]),
    ("fill-changeform-left-den",
       parse_patt @{theory Rational} "a(?b / ?c)/?a = ?b / _",
         [" name1", " name2", " name3" ]),
    ("fill-changeform-none",
       parse_patt @{theory Rational} "a(?b / ?c)/?a= ?b / ?c",
        [" name1", " name2", " name3" ]),
   NONE): fillpat;
*}
```

B.3 Rules Testing for Error-Patterns

The following rules directly step towards a theorem which is related to an error-pattern contained in a list `error_patterns_to_check_` determined by the dialogue.

These rules reside at the side of the *MathEngine*, see Fig.4.6 on p.68.

```
rule "go to a step with an error-pattern given by DG"
  when
    DG#errorPatternsToCheck() != null &&
    intersection (
      ME#fetchProposedTactic().getErrorPatterns(),
      DG#errorPatternsToCheck()) != null
  then
    ME#stepToErrorPatterns (intersection
      (ME#fetchProposedTactic().getErrorPatterns(),
      error\_patterns\_to\_check\_))
    stepping_to_error_pattern_ = true
end
```

In between these rules no user interaction is expected: the former results in a CalcChanged (the ME is expected to produce no error, i.e. no CalcMessage) which in turn actives the following rule:

```
rule "present the step where an EP could occur"
  when
    stepping_to_error_pattern_
  then
    ME#stepOnErrorPattern()        //show fill-form on WS
    fill_pat = ME#getFillpattern()
    WS#showErrorpattern(fill_pat)  //show lhs of fill-pattern
    stepping_to_error_pattern_ = false
end
```

Now the learner can input to the fill-form correctly or not (and no further cases, because the dialogue restricted the situation to this case concerning only one specific EP.)

```
rule "stepping_to_error_pattern: input correct"
  when
    CalcChanced
  then
    stepping_to_error_pattern_ = false
    DG#setEPcounter(err_patt_, 0)
    err_patt_ = ""
    help_counter_ = 0
end

rule "stepping_to_error_pattern: input NOT correct"
  when
    CalcMessage == "stepToErrorPattern: input incorrect"
  then
    stepping_to_error_pattern_ = false
   ....
end
```

B.4 Hint-Pages

Hint-pages are associated with error-patterns (EPs) in a table [1] like the following.

Hint-pages in App.B.4 are within the scope of dialog authors, while error-patterns and fill-patterns in App.B are handled by ISAC's mathematics engine and thus implemented in ML structures are within the scope of mathematics authors.

[1] There are further hint-pages associated to fill-patterns, see App.B.

Also the implementation of the dialog-rules for the dialog guide are still out of scope of dialog authors. But these will come into charge of this implementation task as soon as the language of the DialogGuide has matured (i.e. respective methods have been implemented).

The above table is preliminiarly implemented as a property file named *error_patterns_hint_pages.properties* and located in the file system at *repos/xmldata*.

Appendix C

Deutsche Zusammenfassung

Kognitive Aspekte des Designs von Dialogen in einem Theorem-Beweis Basierten Mathematischen Assistenten
Implemetierung von Fehlermustern zur Steuerung von Dialogen in ISAC

Diese Masterarbeit setzt sich mit den Theorien aus den Kognitionswissenschaften auseinander, die für das Erlernen von Mathematik relevant sind. Die daraus gewonnen Erkenntnisse werden verwendet, um die Erkennung von Fehlermustern auf spezifische Art in das experimentelle System ISAC zu implementieren.
Da es für diese Masterarbeit nötige war, erst diese spezifische Zugangsweise zu finden, beginnt die Arbeit mit einer generellen Beschau des Themas: Die Auseinandersetzung mit den Forschungsergebnissen über die grundlegenden Strukturen und Prozesse des Gehirns im Moment der Anwendung bzw. des Erlernens von Mathematik. Im speziellen wird auf Forschungsergebnisse eingegangen, die sich auf die Abstraktionsfähigkeit und die, in der Anwendung von Mathematik begangenen, Fehler beziehen. Besondere Aufmerksamkeit wurde den Kognitionstheorien geschenkt, die sich spezifisch damit auseinandersetzen, mathematische Lernsoftware zu entwickeln. Im speziellen wird auf Aspekte existierender Software eingegangen, die mit ISAC vergleichbar ist.
ISAC wird dabei als im Entstehen begriffener Prototyp eines auf computerisiertem Theorembeweis basierten Systems gesehen, der als Assistent für das Erlernen von Mathematik agiert — mit groem Schwerpunkt auf den Vorteilen dieses Systems gegenüber anderen. (1) automatisierte Ableitung der Benutzereingabe durch den logischen Kontext und (2) automatische Vorschläge über die nächsten Schritte hin zu einer Lösung in (symbolischen) Rechenbeispielen, ähnlich einer traditionellen stufenweisen Herangehensweise an die Lösung mit Stift und Papier. Diese Dienste eröffnen neue Möglichkeiten einer automatisierten Benutzerführung mittels Dialogen — Das Beweisen der Nützlichkeit dieser neuen Möglichkeiten ist das Hauptanliegen dieser Masterarbeit.
Das konkrete Design von Fehlermustern und deren Implementierung beim Bruchrechnen werden so beschrieben, dass diese Beschreibung als Anleitung für die Weiterarbeit an diesem System aufgefasst werden kann: Der Code, in dem die Implementierung der Fehlermuster vorgenommen wurde, deckt sich mit der Programmiersprache von ISACs Mathematik-Engine; Der Code besteht aus ein paar dutzend Zeilen. Der Code, der die Benutzerführung implementiert, ist in der regelbasierten Spreche eines Expertensystems geschrieben; dieser Code beinhaltet vier Regeln.
Die Implementierung zeigt den generellen Zugang und die Effizienz des Designs: (1) Die Fehlermuster für Brüche werden verallgemeinert und gelten auch für alle anderen Rechenarten, wie Differenzialrechnung, Lösen von Gleichungen, usw. (2) Die Implementierung lässt sich auf alle Berechnungen im Bereich technischer Planung und wissenschaftlicher Arbeit übertragen und (3) die Effizienz zeigt sich dadurch, dass alle unter

(1) und (2) erwähnten Beispiele mit nur vier Regeln abgedeckt werden können. Schlielich kann man aus der Effizienz und allgemeinen Gültigkeit Aussagen über zukünftige Anwendungsbereiche machen: Diese Fortschritte versprechen auf Versuch und Irrtum" aufbauende, unabhängige Lernunterstützung nun auch in der Mathematik, wie es Spiele und Simulationen bereits in anderen Bereichen vorgeben. Diese Art von Software erneuert so die Art des Lernens.

Der Inhalt der Masterarbeit:

1. Kapitel: Stellt allgemeine Begriffe von Kognitionswissenschaften und Mathematik vor. Die führenden Fragestellungen sind: Was ist Mathematik, gibt es angeborene Prozesses, was für eine Rolle spielt Abstraktionsfähigkeit beim Lernen, was ist das Besondere an Algebra uns Bruchrechnungen, was sind die Ursachen für Fehler in der Mathematik, wie kann man Computer im Mathematikunterricht einsetzen.

2. Kapitel: Erklärt die bereits existierenden Funktionen des Theorem-Beweissystems ISAC: Die Hilfe beim nächsten Schritt. Dabei wird auf die Ausgangssituation bei seinem Dialogsystem eingegangen.

3. Kapitel: erläutert ausführlicher den möglichen Beitrag der Kognitionswissenschaften: Wie lösen Schüler Beispiele aus dem Bereich der Bruchrechnungen, welche Art von Wissen braucht man für Mathematik, und was benötigt man für eine effektive Benutzerführung. Das Kapitel vergleicht dazu auch existierende Computerprogramme, die im Mathematikunterricht verwendet werden. Indem wir die eben beschriebenen Elemente zusammenfassen, erstellen wir Anweisungen für ISACs Dialogdesign. Da wir uns auf die Reaktion auf Fehler konzentrieren, gehen wir auch auf die Frage ein, ob Fehler Mustern folgen, und welche Muster nützlich für Dialoge in ISAC sein könnten.

4. Kapitel: Zeigt die konkrete Implementierung von Fehlermustern, Seiten mit Tips, Umformungsregeln, Lückentexten, Dialogregeln und Arten von Dialogen in ISAC.

5. Kapitel: Vergleicht Beispiele aus dem Unterrichtsalttag mit den Möglichkeiten der Fehlererkennung in ISAC. Dieses Kapitel setzt sich auch damit auseinander, wie man die oben beschriebenen Erkenntnisse in das Lernszenario integrieren kann: Vorschläge für geführte Beispiele und Möglichkeiten der künftigen Erweiterung (zB.: Benutzermodelle, Pilotstudien)

Stichwörter: Kognitionswissenschaften, Fehlermuster, Dialog Bruchrechnungen, Lernen, computerisierter Theorem-Beweis, Umformung, Matching, Lukas Interpretation

Appendix D

Curriculum Vitae

Gabriella Daróczy, 2013

Education

2010-2013	Middle European Interdisciplinary Master Programme in Cognitive Science (MEi:CogSci), University of Vienna
2005-2008	M.Sc. in Engineering and Management Faculty of Mechanical Engineering and Information Technology University of Miskolc, Hungary

Work Experience

2012-current	Research Assistant, eMediaMonitoring GmbH, Vienna
2012-current	Volunteering as Mathematical Instructor Lerntafel, Vienna
2010-2012	Private Mathematical Instructor Kogler Karl-Heinz Mag. KEG Nachhilfeinstitute, Vienna
2008-2010	Research and Development in EMCC KIRAS Project Leonardo Internship Rosenbauer International AG, Leonding

Internships, Scholarships, Presentations

2013	Error-Patterns within "Next-Step-Guidance" in TP-based Educational Systems., Gabriella Daróczy, Walther Neuper eJMT the Electronic Journal of Mathematics and Technology Volume 7, number 2 (Special Issue February), pages 175-194
2011 autumn	CEEPUS-Scholarship Comenius University of Bratislava Measuring the EEG mu rhythm desynchronization associated with motor actions Supervisor: Prof. Igor Farkas
2011 autumn	Semester Project: Nature of Human Consciousness and Narrativity Institute of Philosophy, Slovak Academy of Science, Bratislava Supervisor: Prof. Silvia Gáliková
2011 spring	Learning effects in Reckon and Choose (L4S) OFAI, Vienna (Austrian Institute of Artificial Intelligence) Supervisor: Dr. Paolo Petta
2011-2012	Science and Art: Effect of Different frames and alignments of a painting on the distribution of viewing directions in human perception. Laboratory for Empirical Image Science, Department of History of Art University Vienna Supervisor: Prof. Raphael Rosenberg
2007	TDK (Scientific Students' Associations Conference University Miskolc) I. Place, Lifetime Management of Steel Bridges Supervisor: Dr. János Lukács
2008 autumn	DAAD-Scholarship, Heinrich Heine University, Düsseldorf, Germany
2005 spring	Erasmus-Scholarship, Tampere University of Technology, Tampere, Finland
2005, 2007, 2008	Scholarship of the Hungarian Republic
2003-2005	Member of Mathematical Self-Education Circle University of Miskolc, Hungary
2004	Remarks in the History of Mathematics (Presentation) 6th Junior Mathematical Congress Stockholm (23.-28. Juni)