

A Wizard-of-Oz Experiment for Tutorial Dialogues in Mathematics

Christoph BENZMÜLLER¹, Armin FIEDLER¹, Malte GABSDIL², Helmut HORACEK¹,
Ivana KRUIJFF-KORBAYOVÁ², Manfred PINKAL², Jörg SIEKMANN¹,
Dimitra TSOVALTZI², Bao Quoc VO¹, Magdalena WOLSKA²

¹*Department of Computer Science* ²*Department of Computational Linguistics*
Saarland University, P.O. Box 15 11 50
D-66041 Saarbrücken, Germany

Abstract. In this paper we report on a Wizard-of-Oz (WOz) experiment which was conducted in order to collect written empirical data on mathematics tutorial dialogues in German. We present a methodological approach for optimising the gains from WOz empirical studies. We show the results of this approach from our empirical study.

1 Introduction

In a Wizard-of-Oz (WOz) experiment, the subject interacts through an interface with a human “wizard” simulating the behaviour of a system [1]. The WOz methodology is commonly used to investigate human-computer interaction in systems under development.

In this paper we report on a WOz experiment which was conducted in the framework of the DIALOG project [2] in order to collect empirical data on mathematics tutorial dialogues in German. More specifically, our goal was to collect data on (1) the tutoring process, (2) the students’ answers, (3) the dialogue behaviour, and (4) the use of natural language.

The reason for using the WOz methodology is that we can formalise the model which we want to implement in our system and ask the wizard to follow it. This way (i) dialogue data which represents the users’ behaviour in interactions following the specific model can be collected and (ii) an early feedback on the model is provided. In subsequent experiments in the project, implemented components can be substituted for some of the tasks now carried out by the wizard, while preserving the overall experimental setup.

In the DIALOG project, we aim at a mathematical tutoring dialogue system that employs an elaborate natural language dialogue component. Our motivation stems from empirical evidence that natural language dialogue capabilities are necessary for the success of tutoring [3]. Moreover, to model mathematics tutorial dialogues, we need a formally encoded mathematical theory, means of evaluating the student’s input in terms of the knowledge of the domain demonstrated, and a theory of tutoring.

In this paper, we first show how the preparations for our experiment addressed these issues and we look at the formalisations which enable us to specify the goals of our experiment more robustly in Section 2. Then, in Section 3, we present the actual experiment design which made use of the preparation. Next, we discuss our approach in Section 4. In Section 5 we look into some related work and then conclude the paper.

2 Methodology and Formalisations

Our general motivation was to formalise the different areas of interest in order to restrict the collection of the data. The formalisations secure a consistent behaviour from the wizards. It thus becomes possible to evaluate this behaviour in general, and to extract robust qualitative information from the data on how to improve it.

To this end, we enhanced an existing mathematical ontology used by the theorem prover Ω MEGA [4] by making explicit relations, which can be used in the tutoring. Moreover, to classify the student's input, we developed a categorisation scheme for student answers, which draws on the mathematical ontology. We further chose to implement the *socratic* [5, 6] teaching strategy and, with it, hinting. For this reason, we developed a taxonomy of hints for the naive set theory domain, which is also based on the mathematical ontology. This taxonomy is used by a hinting algorithm which models the socratic tutoring method by means of producing different hints according to an implicit student model [7].

Domain Ontology To automate the tutor's behaviour, we structured the naive set theory domain. The structured ontology can be used for evaluating the student's input and in automating the hints of our taxonomy. We made use of the domain ontology of the proof planner Ω MEGA and enhanced it for the further needs of tutoring.

Ω MEGA makes use of a mathematical database that is organised as a hierarchy of nested mathematical theories. Each theory includes definitions of mathematical concepts, lemmata and theorems about them, which we will collectively call *assertions* henceforth, and inference rules, which can be seen as lemmata that the proof planner can directly apply. Moreover, each theory inherits all assertions and inference rules from nested theories.

Here are a few examples of definitions of the mathematical concepts in intuitive terms as well as in more formal terms paraphrasing but avoiding the λ -calculus formulae as they are represented in Ω MEGA's database:

Let U, V be sets and let x be an inhabitant.

- The *elements* of a set are its inhabitants: $x \in U$ if and only if x is an inhabitant of U .
- A set is a *subset* of another set if all elements of the former are also elements of the latter: $U \subseteq V$ if and only if for all $x \in U$ follows that $x \in V$.

Enhancing the Ontology Ω MEGA's mathematical database implicitly represents many relations that can be used in tutorial dialogues. Further useful relations can be found when comparing the definitions of concepts with respect to common patterns. We consider relations between mathematical concepts, between mathematical concepts and inference rules, and among mathematical concepts, formulae and inference rules. By making these relations explicit we enhance the mathematical database such that it can be used in hinting.

Again, we only give a few examples of relations between concepts, which mostly make use of the definitions we have already given.

Antithesis: σ is in *antithesis* to σ' if and only if it is its opposite concept (i.e., one is the logical negation of the other).

Hypotaxis: σ is in *hypotaxis* to σ' if and only if σ' is defined using σ . We say, σ is a *hypotaxon* of σ' , and σ' is a *hypertaxon* of σ .

Examples: \in is a hypotaxon of $\subseteq, \supseteq, \cup$

Primitive: σ is a *primitive* if and only if there is no hypotaxon of σ .

Examples: \in is a primitive

Note that \in is a primitive in Ω MEGA's database, since it is defined using *inhabitant*, which is a type, but not a defined concept.

Using the Domain Ontology The ontology we presented is evoked, among other things, in categorising the students' answers. That is, the algorithm takes as input the analysed student answer. In analysing the latter, we (i) compare it to the expected answer and (ii) look for the employment of necessary concepts. These necessary concepts are defined in terms of the ontology. The algorithm checks for the student's level of understanding by trying to track the use of these concepts in the student answer to be addressed next. The hint to be produced is then picked according to the knowledge demonstrated by the student. Note that this knowledge might as well have already been provided by the system itself, in a previous

hinting turn when dealing with the same proof step. Since the algorithm only checks for generic descriptions of those concepts, we suggest the use of the present ontology in order to map the descriptions onto the actual concepts relevant to the particular context.

For more on the enhanced ontology see [8].

Student Answer Evaluation In order to evaluate the student's input, we define student categories based on their completeness and accuracy with respect to the expected answer. The *expected answer* is the proof step which appears next in the formal proof for the problem at hand. These categories are input for the implicit student model of the hinting algorithm.

We define completeness and accuracy with regard to the parts of an answer. In general terms, *parts* are the premises, the conclusion and the inference rule of a proof step.

We define the predicates complete and accurate as follows:

- An answer is *complete* if and only if all parts of the expected answer are mentioned.
- A part of an answer is *accurate* if and only if the propositional content of the part is the true and expected one.

Completeness From our definition of completeness it follows that completeness is dependent on domain objects, but not on our domain ontology. That is, the expected answer, which is the basis of the evaluation of the completeness of a student answer, necessarily makes use of objects in the domain. However, the relations of the objects in the domain are irrelevant to evaluating completeness. Moreover, a place holder for an expected object in the answer is enough for attributing completeness, no matter if the object itself is the expected one.

Accuracy Accuracy, contrary to completeness, is dependent on the domain ontology. It refers to the appropriateness of the object in the student answer with respect to the expected object. An object is accurate, if and only if it is the exact expected one.

The Categories In this section we enumerate the categories of students answers based on our definitions of completeness and accuracy and with regard to the expected answer.

We define the following student answer categories:

Correct: An answer which is both complete and accurate.

Complete-Partially-Accurate: An answer which is complete with some inaccurate parts.

Complete-Inaccurate: An answer which is complete, but all parts in it are inaccurate.

Incomplete-Accurate: An answer which is incomplete, but all parts in it are accurate.

Incomplete-Partially-Accurate: An answer which is incomplete and some inaccurate parts.

Wrong: An answer which is both incomplete and inaccurate.

For the purposes of the experiment, we collapsed the categories complete-partially-accurate, complete-inaccurate and incomplete-partially-accurate to one category, namely, inaccurate. Nevertheless, we expected the experiment to show us if and how these categories should have their own separate place as well as if and how certain categories should be further refined.

We also wanted to cover the possibility of over-answering and extract information on formalising it later on. We defined (accurate or inaccurate) over-answering as several distinct answers. That is, if the student's answer included more proof steps than one, we considered the steps as multiple answers. The categorisation would then be applied to them separately.

More on our student answer categorisation scheme can be found in [9].

Teaching Strategies For the purposes of the experiment, we define not only an eliciting tutoring strategy that we propose for our system, but also two more strategies; an explanatory and a minimal feedback strategy. We shall follow Person and colleagues [5] and Rosé and colleagues [6] in calling the eliciting strategy *socratic* and the explanatory strategy *didactic*. The didactic strategy, as we will see, is called within the socratic strategy itself. Therefore, there is as much need to define it in order to collect qualitative data that help us test and

enhance it as there is for the socratic strategy. We also define a minimal feedback strategy as a control group strategy for a comparative evaluation of the three strategies.

In formalising the teaching strategies, and especially the hinting process, we aim at restricting the wizards' behaviour to what can be modelled. Taking these restriction into account, we collect useful data that can help us improve and extend our formalisation.

Socratic Strategy In DIALOG, we aim at a mathematical tutoring system that tutors proving a theorem in a way that helps the student understand the current proof, and allows for a high learning effect. What is meant by the latter is the ability of the students to better understand the problem at hand, as well as to generalise and apply the taught strategies alone later on.

Based on psychological evidence [10, 6] for the high educational effect of hinting, we propose to establish those tutoring aims by making use of the socratic tutoring method. The decisive characteristic of the socratic strategy is exactly the use of hints in order to achieve self-explanation [6, 11]. We, therefore formalised the hinting process building on the, little, systematic research done to date in the area [12, 13, 14].

A Hint Taxonomy In order to model hinting we first need a taxonomy of hints. We defined hint categories based on the needs in the domain, as they revealed through the ontology.

The hint taxonomy captures the underlying function of hints that can be common for different surface realisations. This underlying function is mainly responsible for the educational effect of hints. The structure of the hint taxonomy also reflects the function of the hints with respect to the information that the hint addresses or is meant to trigger. In order to capture the different functions of a hint in the taxonomy we defined hints across two dimensions.

The first dimension distinguishes between the active and passive function of hints. The former refers to the information provided each time and the latter to the information that the hint aims at triggering in the student's current cognitive state, that is, the information elicited.

The second dimension distinguishes between different classes of hints. Each of these classes consists of single hint categories that elaborate on one of the attributes of the proof step under consideration. The hint categories are grouped in classes according to the kind of information they address in relation to the domain and the proof. By and large, the hints of the passive function of a class in the second dimension constitute the hints of the active function of its immediately subordinate class, in the same dimension. For example, the passive hint *give-away-antithesis* of the class *domain-relation* is also an active hint of its subordinate class, namely, *domain-object*. In providing this hint the system is trying to elicit the object in the domain which would help the student proceed with the proof. Both the relation *antithesis* and the object elicited are defined through our domain ontology (cf. Section 2).

For the complete hint taxonomy and a discussion of the hint categories in it see [7].

A Hinting Algorithm A tutorial system ideally aims at having the student find the solution to a problem alone. Only if the student gets stuck should the system intervene. Based on [14] we derived a socratic algorithm that implements this. We aimed at a user-adaptive algorithm, which chooses hints tailored to the students. If hints do not help the algorithm switches to an explanatory strategy (see didactic method), it gives away the answer and explains it.

In intuitive terms, the algorithm aims at having the student find the proof by himself. If the student does not know how to proceed or makes a mistake, the algorithm prefers hinting at the right solution in order to elicit the problem solving instead of giving away the answer.

An implicit student model makes the algorithm sensitive to students of a different level by providing increasingly informative hints. The algorithm takes into account the current and previous student answers. The input to the algorithm is the category that the student answer has been assigned, based on our student answer categorisation scheme (cf. Section 2), and other relevant domain knowledge that the student might possess. Moreover, the algorithm computes whether to produce a hint and which hint to produce, based on the number of

wrong answers, as well as the number and kind of hints already produced.

Finally, the algorithm takes as input the analysed student answer, which determines which parts of it are problematic and need to be addressed, and how they are going to be addressed.

For a more detailed discussion of the hinting algorithm see [7].

Didactic Strategy We defined the didactic strategy as an explanatory method of teaching. In this method, the students' answers still have to be categorised. However, hints are not provided. That means that whenever the student makes a mistake, the system gives the correct answer away together with an explanation. Correct answer here refers to the reasoning for arriving at the expected performable step, or the step itself, if the student possesses all the reasoning steps but not the proof step. In the end, the whole proof is summarised.

Minimal Feedback Strategy *Minimal feedback* was defined to model non-tutoring. It gives answers that students can more or less find on their own in a textbook. It also requires categorising the student answer, but there are no hints and no explanations in this strategy. The student is informed of the approximate degree of correctness of his answer, with no pointers to what exactly is correct or wrong, in case of partially correct answers. In effect, the system accepts correct answers but when the answer is wrong, the student is simply informed of it and prompted for the correct answer. The whole proof is only given in the end.

Subdialogues Although, we did not know what forms of subdialogues are necessary for tutorial dialogues in mathematics and consequently could not formalise subdialogues, we wanted to collect relevant data. Therefore, we allowed for the use of subdialogues in the algorithms for the socratic and the didactic strategies.

DiaWoz: A Wizard-of-Oz Tool We implemented a tool called DiaWoz [15] to support the experiment and collect dialogue data on-line. This tool has an interface for the subject. It enables the subject to type text or insert mathematical symbols by clicking on buttons; it also displays the complete dialogue with both the tutor's and the subject's utterances. The wizard's interface features two additional drop-downs; one for categorising the subjects input and one for the hint produced, necessary for the socratic strategy. This annotation appears in the electronically collected corpus and facilitates its analysis. Although we used a version of DiaWoz with additional features for the particular purposes of this experiment, DiaWoz is a generic tool that can be easily adapted for the special needs of any WOz experiment.

Section 3 presents the experiment design and explains the use of the formalisations.

3 Experiment Design

The aims of the design were (i) to facilitate the collection of useful data for the formalisations and of unbiased linguistic data and (ii) to check the change in performance tutoring effects.

After the formalisations we were able to define the original goals of the experiment more precisely. In particular, we tested and collected qualitative data on:

- (1) The sufficiency and effectiveness of the formalised hinting categories.
- (2) The appropriateness of the domain ontology for the automatic production of hints and the categorisation of the student's answer.
- (3) The drawbacks and possible improvements of the hinting algorithm.
- (4) The applicability of the student answer categories and ways to refine it.
- (5) The use of subdialogues.
- (6) The dialogue behaviour of the student and the tutor in the defined tutoring context.
- (7) The use of natural language for realising the hints.
- (8) The use of natural language in the student's answers.

In this paper we do not look into the last three points, which make up a self-contained issue.

The experiment consisted of three phases. In the first pre-tutoring phase information about the original level of the student was collected. The second phase comprised the tutoring based on the three pre-defined strategies whose learning effect we wanted to test. The third phase facilitated the evaluation of the student level after tutoring. Two questionnaires, two test-proofs and the overall experiment were designed to enable the evaluation.

Hypothesis Our null hypothesis was: The performance change of the students in the three groups would not differ significantly. That would in turn mean that our formalisable hint categories, the hinting algorithm and the student answer evaluation scheme are not adequate. The qualitative data could be used for improving them.

People and Environment The people involved in the experiment were: (i) the experimenter (ii) the subjects (iii) the wizards.

The experimenter answered questions relevant to the experiment, for example, on the questionnaires and the interface. He was the only person the subjects had contact with.

In total, 24 subjects participated in the experiment. They were students with a humanities or science background. Their prior mathematical knowledge ranged from little to fair.

There were three wizards. The tutor was a holder of a masters in mathematics with experience in tutoring. He was responsible for the keyboard communication with the subjects, for categorising the student input (cf. Section 2) or initiating a subdialogue, and for verbalising the hinting categories¹. The assumed tactic with its pedagogical ramifications was explained to the tutor in the form of instructions as to what he was expected to do. He was then trained on the task. The tutor's assistants were the developers of the formalisations and the algorithm. They were responsible for applying the hinting algorithm and, more generally, restricting the tutor's behaviour to what can potentially be modelled.

Two adjacent rooms with a one-way window were used. The subjects were in one room, the wizards and the experimenter in the other with the ability to see the subject. Communication between the experimenter and the subjects was possible via headphones and microphones. The wizards could see the student's DiaWoz window as well as their own. The student could only see the latter. Each experiment lasted approximately two hours.

Data was collected by videotaping both rooms², by questionnaires, in form of notes by everybody involved and electronically by DiaWoz. The electronically collected data constitutes the corpus on tutorial dialogues in mathematics.

Phase 1: Before Tutoring

Preparation We first gave the subjects a questionnaire for personal details and mathematical knowledge. We asked them to give definitions of some domain concepts and assess their own level. We then gave them written instructions on the experiment. These included a description of the phases of the experiment and what they would be asked to do. The subjects were lead to believe that they would be evaluating a tutoring dialogue system.

Lesson Material Subjects read a lesson material without a time limit. It included all domain knowledge needed for the tasks and some extra material. Domain knowledge comprised an introduction to the naive set theory, definitions of concepts, theorems and lemmata.

Pre-test After having read the lesson subjects did a timed pre-test for the task, that is, they attempted a proof (cf. Figure 1: Proof 1)³.

Phase 2: Tutoring Subjects were split into three groups, one for every defined teaching

¹Since we were aiming at the collection of unbiased linguistic data, verbalisation was totally free.

²Subjects were asked and encouraged to think aloud whenever they were performing a proof task.

³Henceforth, K stands for the complement of a set and P denotes the powerset of a set.

- (1) $K(A) \in P(K(A \cap B))$
- (2) $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$
- (3) $A \cap B \in P((A \cup C) \cap (B \cup C))$, where P denotes the power set of a set
- (4) if $A \subseteq K(B)$ then $B \subseteq K(A)$
- (5) $K(A \cup B) \in P(K(A))$

Figure 1: Experiment Proofs

strategy: socratic, didactic and minimal feedback. Subjects were assigned to different strategies at random and the order of strategies was also random. The tutor's behaviour modelled the defined teaching strategy according to the group each subject belonged to.

Dry-Run Subjects first got technical instructions on how to use the interface and the tutoring system. They then did a dry-run (cf. Figure 1: Proof 2) on an easy proof to gain familiarity with the interface and the tutoring.

Main Task All students were confronted with two more difficult proving tasks (cf. Figure 1: Proofs 3 and 4) presented in a randomised order so that we could avoid any possible correlation between the order of presentation and the learning effect. Due to time restrictions, subjects could run out of time during an attempt, whereupon the session would be terminated.

Wizards: Tutor and Assistants The tutor first classified the subject's contribution. A standard time-out for responding to the system's utterances was used, when the subject remained totally idle. Then, the wizard decided what dialogue moves to make next and verbalised them. Depending on the tutoring strategy employed by the wizard for a given subject, the obligatory dialogue moves included informing the subject about completeness, accuracy, and relevance of the utterance, giving hints on how to proceed further, explaining a step under consideration, prompting for the next step, or entering into a clarification dialog. The tutor was also free to realise any other dialogue moves he wished to and to mix text with formulae. At the end of every session, and irrespective of the performance of the subject the tutor presented the answers to the proofs on the computer screen. The classification of the contribution and the hint category produced were the only things annotated electronically on the spot.

For the socratic strategy in particular, the assistants told the tutor which hint to realise each time. The hint was chosen based on the algorithm but the way to realise it was left totally on the tutor. Moreover, although it was the tutor's decision to produce a hint at all or initiate a subdialogue, his behaviour was restricted by the assistants to what can be modelled.

Phase 3: After Tutoring

Post-test: Subjects were asked to attempt on paper a final proof (cf. Figure 1: Proof 5). This proof was a variant of the original pre-test proof and it involved using domain concepts also included in the tutored proofs.

Prior to the experiment, the pre-test and post-test were tested for a possible ceiling effect (i.e., if they were too easy) as well as for ensuring that their degree of difficulty was equivalent. This would allow us to evaluate the performances in the two proofs, and then compute a learning effect attributed to the tutoring that mediated the two attempts.

Questionnaire The subjects were asked to fill in a questionnaire addressing various aspects of the system, including both the interface and the tutoring and its usability. The questionnaire included both questions which asked the subjects to rate specific attributes of the system in a scale, and open questions, which asked for descriptive answers.

- T1:** Bitte zeigen Sie: Wenn $A \subseteq K(B)$, dann $B \subseteq K(A)$! [*Please prove: If $A \subseteq K(B)$, then $B \subseteq K(A)$!*]
S1: $K(B) = U \setminus B$
T2: Das ist richtig, aber wie geht es weiter? [*That is correct, but how do you want to continue?*]

Figure 2: Correct irrelevant answer, Subject 13

- S5:** wenn $A \subseteq K(B)$, dann $A \neq B$, weil $B \neq K(B)$ [*if $A \subseteq K(B)$, then $A \neq B$, because $B \neq K(B)$*]
T6: meinen Sie wirklich \neq oder etwas anderes? [*Do you really mean \neq or something else?*]
S6: \neq

Figure 3: Subdialogue, Subject 13

4 Findings

In this section we first present some qualitative data that validates the general methodology used. We then consider a few points for improvement.

Results on Formalisations As expected, we collected valuable data on the formalisations. We present here some indicative cases with relation to the preparations of the experiment.

Student Answer Evaluation: We have already spotted problematic or insufficient areas in the categorisation scheme that was used. We will mention only a few. First, a more formal definition of parts was necessary. This issue has already been dealt with in [9]. Second, the data reveals specific needs for building and using a representation that enables matching the students' contributions to the proof to an expected contribution. In particular, handling of implicit inference steps, over-answering, correct irrelevant answers, starting the proof from scratch and choosing the expected (intended) proof step are all issues that can be addressed by use of the data. In Figure 2⁴, for example, the student says something right, but the tutor cannot match it to any proof. He therefore, asks him to show what he is getting at. After a couple of turns it actually turned out that the subject did not know how to continue.

Moreover, the refinement of the categories is possible. We have identified possible answers that can be considered as near-misses (bracketing problems, use of a ' \in ' instead of ' \subset ') and were thus treated by different kinds of subdialogues. This information will also help us model subdialogues themselves, whose use was allowed in the experiment, but were not formalised. In the example in Figure 3, the tutor could not make sense of the student's utterance unless he substituted \cap for \neq . He tried to elicit a self-correction. The student could not provide it, although he realised from the question that he had used the wrong symbol.

In addition, we are considering the sub-division of the category 'wrong' into two categories, namely 'wrong attempt', when the student gives a wrong answer, and 'no attempt', when the student gives up without any attempt. Finally, we are looking into formalising misconceptions that have a yet special status, and should be addressed accordingly.

Hinting: It is now possible to include in the hinting algorithm defined hint categories that we were not able to include before, as we did not know how they should be managed in the specific domain. An example of this is the category *confer-to-lesson*. For instance, in Figure 4, the student has mixed up two rules. The tutor tells him to look them up.

We are also considering the definition of hints that would prove useful. For example, there seems to be a formalisable category for explicitly connecting the current state of the proof to something mentioned before. Such a hint requires collaboration between discourse structure and task management. For that, we will use the video data, which includes the students' thinking aloud, and the tutor's notes on points of disagreement with the instructed behaviour.

The same data will be used for enhancing the hinting algorithm. For one, the implicit student model has to be enhanced. We are investigating modifying that to take into account

⁴All examples include a translation from German in italics, where necessary.

- T4:** Meinen Sie wirklich: $P(C \cup (A \cap B)) = P(C) \cup P(A \cap B)$?
 [Do you really mean: $P(C \cup (A \cap B)) = P(C) \cup P(A \cap B)$?]
- S4:** ich denke doch: $P(A) \cup P(B) = P(A \cup B)$ [yes, I think so: $P(A) \cup P(B) = P(A \cup B)$]
- T5:** Das ist nicht richtig! Vielleicht sollten Sie noch einmal in Ihrem Begleitmaterial nachsehen. [That is not correct! Maybe you should look it up in your study material.]

Figure 4: Confer-to-lesson, Subject 20.

- (1) $A \in P(A \cup C), B \in P(B \cup C) \quad A \cup B \in P(A \cup C) \cap (B \cup C)$
 (2) $P(A \cap B), P(C) \subseteq P((A \cap B) \cup C)$

Figure 5: Different uses of comma.

subdialogues, which were initially excluded from the implicit student model, as they had not even been formalised. Moreover, augmenting the algorithm to better accommodate students' answers is facilitated by the data, by using such instances as the basis for our research.

Student Model: Some of the aspects of student modelling became obvious, both with regard to hinting and to the external student model that should be responsible for the choice of the right level of lesson material for the student and the degree of difficulty of the task. The significance of the latter was demonstrated in the experiment by the variance of the performance and the degree of satisfaction. A subject specifically commented that the system is for mathematicians. It is already possible to extract information for student modelling in our domain. A simple example would be identifying relevant questions and the conclusions that can be drawn about the student's level, which can be used for building the student model. Incorporating a student model would also permit different levels of abstraction for the same proof to be taken into account. Unnecessary frustration would be then avoided. The system, for instance, could judge when the level of the student is high enough to allow implicit proof steps, or significantly low to assume that there was a mistake if a step is missed out.

All the above observations have consequences for the system requirements and architecture, which we do not discuss in this paper. For more on this issue see [16].

Discussion of Experiment Design On the whole we found that our experiment design was adequate. However, there are points where better planning would have proven beneficial.

Natural language processing was not restricted to any realistic extent in this pilot experiment and was too good. For example, although the use of comma was ambiguous, it was used to mean both logical "and" and to just separate elements in a list (cf. Figure 5), the system always interpreted it right in the given context. This was due to the desire to collect free linguistic data. In subsequent experiments, though, we will formalise this aspect of the system and restrict it by help of the data from this study, in order to collect more useful data.

The dry-run and proof sessions were much longer than expected. Therefore the subjects did not have in-between breaks. As a result, the possible deterioration of the students' performance due to fatigue was not controlled. Moreover, different subjects were tutored on a shorter or longer part of the proof depending on the group they belonged to. On the whole, the minimal feedback group saw the whole proof in the predefined amount of time, the didactic saw a fair amount, and the socratic hardly ever managed to move beyond the first two proof step. There was, in effect, a disadvantage for the socratic group.

Some subjects had a much better level both at the task and at using the interface. They were normally the ones that stated in the questionnaire that their knowledge in mathematics was very limited. This effect became worse by the time restriction, as subjects of a better level had more time to progress through the proofs. In addition, the dry-run proof, which was intended to be very easy to allow equal familiarity with the interface, was so difficult for some subjects that they hardly used the system and did not gain familiarity. Unfortunately,

the small numbers of the subjects do not help to even out any differences between groups.

One of the effects of the socratic teaching strategy, as opposed to other strategies, is that such cognitive faculties are stimulated which enable a learning effect over time. In other words, the students actually learn and do not simply memorise. This, however, was never tested since subjects were only asked to show what they had learned right after tutoring.

A major aim of tutoring was to teach proving techniques. Nevertheless, the test proof was not well chosen to test if the student learned how to tackle the proving problem. It rather concentrated on the definitions of domain concepts, which are by no means the only relevant thing. The assumed aim of the tutoring tool is to practice already taught material, and not to teach from scratch. Moreover, although the subjects were given definitions of concepts, they were not told how one works with a proof. It is characteristic that Subject 26 explicitly communicated to the experimenter that they did not know what a proof was.

5 Related Work

Rosé et al. [17] have conducted WOz experiments with similar aims for the domain of Basic Electricity and Electronics. They have collected a corpus of tutorial dialogues and did a comparative evaluation of the socratic and the didactic strategies. Although they found a tendency for the socratic strategy, they had not formalised the two strategies before the experiment, neither did they have a control group like our minimal feedback group. It is therefore difficult to know what exactly their socratic strategy is, and most importantly, if the human behaviour of their tutor can indeed be modelled. We have, nevertheless, extracted valuable information from their corpus for the formalisation of our hinting process.

We also found the discoveries of other relevant work very useful in the preparation of our WOz empirical study. Maulsdy et al. applied the design for the instructible agent Turvy [18]. They also based their experiment on a priorly defined formal model of the agent they wanted to build, which the wizard had to follow. They found this very useful for the collection of qualitative data, which in turn proved more valuable for the evaluation and improvement of Turvy. The importance of the latter is also pointed out by Pirker et al. [19], in a study for different design possibilities for a speaking dialogue system.

Salber et al. [20] used the WOz technique to define general multimodal interfaces requirements. We followed their method in having multiple wizards, namely three, in order to distribute the cognitive load. In a demanding study like ours, where the tasks which had to be performed by the wizard were many and difficult, this proved a sine qua non.

6 Conclusions for Future Research

We proposed a methodology for a WOz experiment which we used to collect a corpus of tutorial dialogues in mathematics in German. We presented the experiment preparations in order to collect this data. More specifically, we presented the formalisation of hinting in order to model the socratic teaching strategy. This involved developing (i) an enhanced domain ontology to be used in the automation of the hinting process, (ii) an evaluation scheme for categorising the student answer, (iii) a hint taxonomy with multiple hint categories, and (iv) a preliminary algorithm which makes use of the above for selecting the system's dialogue moves. We also defined two additional counter-strategies, didactic and minimal feedback, to enable a comparative evaluation.

We showed how using the particular methodology provided us with good qualitative data. These data will be used for the augmentation of our model towards the final implementation of a system. We also reported on the valuable experience gained from the experiments.

In the future, we plan to conduct additional WOz experiments both to further evaluate the augmented model and to collect additional dialogue data. To this end, we will concentrate on formalising a dialogue theory, informed by the existing corpus and from pedagogical theories.

References

- [1] N. Bernsen, H. Dybkjær, and L. Dybkjær. *Designing Interactive Speech Systems — From First Ideas to User Testing*. Springer, 1998.
- [2] M. Pinkal, J. Siekmann, and C. Benz Müller. Projektantrag Teilprojekt MI3 — DIALOG: Tutorieller Dialog mit einem mathematischen Assistenzsystem. In *Fortsetzungsantrag SFB 378 — Ressourcenadaptive kognitive Prozesse*, Universität des Saarlandes, Saarbrücken, Germany, 2001.
- [3] J. Moore. What makes human explanations effective? In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pp. 131–136. Hillsdale, NJ. Earlbaum, 2000.
- [4] J. Siekmann *et al.* Proof development with Ω MEGA. In Andrei Voronkov, ed., *Automated Deduction — CADE-18*, number 2392 in LNAI, pp. 144–149. Springer Verlag, 2002.
- [5] N. Person *et al.* Dialog move generation and conversation management in AutoTutor. In C. Rosé and R. Freedman, eds., *Building Dialog Systems for Tutorial Applications—Papers from the AAAI Fall Symposium*, pp. 45–51, North Falmouth, MA, 2000. AAAI press.
- [6] C. Rosé *et al.* A comparative evaluation of socratic versus didactic tutoring. In Johanna Moore and Keith Stenning, eds., *Proceedings 23rd Annual Conference of the Cognitive Science Society*, University of Edinburgh, UK, 2001.
- [7] A. Fiedler and D. Tsovaltzi. Automating hinting in mathematical tutorial dialogue. In *Proceedings of the EACL-03 Workshop on Dialogue Systems: interaction, adaptation and styles of management*, pp. 45–52, Budapest, 2003.
- [8] D. Tsovaltzi and A. Fiedler. Enhancement and use of a mathematical ontology in a tutorial dialogue system. In *IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico, 2003.
- [9] D. Tsovaltzi and A. Fiedler. An approach to facilitating reflection in a mathematics tutoring system. In *Proceedings of AIED Workshop on Learner Modelling for Reflection*, Sydney, Australia, 2003.
- [10] M. Chi *et al.* Eliciting self-explanation improves understanding. *Cognitive Science*, 18:439–477, 1994.
- [11] D. Tsovaltzi and C. Matheson. Formalising hinting in tutorial dialogues. In *EDILOG: 6th workshop on the semantics and pragmatics of dialogue*, pp. 185–192, Edinburgh, UK, 2002.
- [12] G. Hume *et al.* Hinting as a tactic in one-on-one tutoring. *Journal of the Learning Sciences*, 5(1):23–47, 1996.
- [13] A. Fiedler and H. Horacek. Towards understanding the role of hints in tutorial dialogues. In *BI-DIALOG: 5th Workshop on Formal Semantics and Pragmatics in Dialogue*, pp. 40–44, Bielefeld, Germany, 2001.
- [14] D. Tsovaltzi. Formalising hinting in tutorial dialogues. Master’s thesis, The University of Edinburgh, UK, 2001.
- [15] A. Fiedler and M. Gabsdil. Supporting progressive refinement of Wizard-of-Oz experiments. In C. Rosé and V. Aleven, eds., *Proceedings of the ITS 2002 — Workshop on Empirical Methods for Tutorial Dialogue Systems*, pp. 62–69, San Sebastián, Spain, 2002.
- [16] C. Benz Müller *et al.* Tutorial dialogs on mathematical proofs. In *IJCAI Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, Acapulco, Mexico, 2003.
- [17] C. Rosé *et al.* A comparative evaluation of socratic versus didactic tutoring. In *Proceedings 23rd Annual Conference of the Cognitive Science Society*, Edinburgh, UK, August 2001.
- [18] D. Mulsby *et al.* Prototyping an intelligent agent through wizard of oz. In *ACM SIGCHI Conference on Human Factors in Computing Systems*, Amsterdam, The Netherlands, May 1993.
- [19] H. Pirker *et al.* Thus spoke the user of the wizard. In *The 6th European Conference on Speech Communication and Technology (Eurospeech99)*, Budapest, Hungary, 1999.
- [20] D. Salber and L. Coutaz. Applying the wizard of oz technique to the study of multimodal systems. In *EWHCI*, Moscow, Russia, 1993.