Probability estimation and competence models for rule and strategy based e-tutoring systems

A University of Utrecht Computing Science master thesis
Author: Diederik Marijn Roijers (student ID: 0310565)
Supervisors: prof. dr. Johan Jeuring and dr. Ad Feelders
Thesis ID: ICA-0310565
June 5, 2011

Universiteit Utrecht

Contents

1 Introduction

1.1 Teaching and learning, strategies to solve problems

Students of natural sciences have to learn how to solve many types of standard problems. Usually, these problems should be solved by rewriting some kind of formula or expression, step by step, until the problem is solved. Each domain (e.g. algebra, matrix calculus or logic) has its own set of applicable rewrite rules. Students learn how to combine these rules into a solving strategy.

Students start studying this type of problems early in their school carreers. For example, students learn how to solve quadratic equations: they learn that $x^2 + (a+b)x + ab = 0$ can be rewritten into $(x+a)(x+b) = 0$ and that $(x + a)(x + b) = 0$ can be rewritten into the solution of the problem, which is $x = -a \vee x = -b$.

For learning rewrite rules and solving strategies, a lot of input from teachers is usually required. The teacher provides scaffolding - sufficient support to acquire the understanding of rewrite rules and others concepts in the domain and the skills to solve the problems in the domain. The teacher provides instruction, exercises, and then feedback on how a student is solving these exercises. This process is of course a very labor-intensive process, which requires a lot of (individual) attention from the teacher. As there is usually only one teacher per classroom, students will often have to wait until the teacher is available. When they work on problems outside of class hours, they will often have to wait until next class if they cannot solve an exercise by themselves. E-tutoring systems are programs which support the teaching and learning process by automatically providing feedback, thus making this process more efficient.

1.2 Rule and strategy based e-tutoring systems

The Open University in the Netherlands has created an e-tutoring system based on rules and strategies: `ideas` [GHJ+07]. Heeren et al. [HJG10] define how to use rewrite rules and strategies in e-tutoring systems to provide feedback. They define how to combine rules into strategies using strategy combinators, which is are similar to parser combinators. Using these strategy combinators they define complete strategies, in order to solve a variety of problems in different domains.

Succesful implementations of strategies are available for bringing a logic proposition to disjunctive normal form and reducing a matrix [HJG10], for

high school mathematics [JH09] and even for functional programming exercises [GJH10].

In `ideas` a student solves a problem step by step. Each step is an application of a rule. After a step, the students submits his/her intermediate solution, and is provided with feedback by `ideas`. This feedback can be that the student has applied the rule correctly. Alternatively, the feedback can be that a mistake has been made, which can in turn be the result of a known error, a buggy rule [GHJ08]. The `ideas` environment can also provide feedback on whether or not the preferred strategy, which is embedded in the system, is being followed by the student.

A rule and strategy based e-tutoring system provides the student with direct feedback. This can accellerate the learning process as more individual feedback can be provided, anytime, anywhere. The teacher only has to define the rules, the strategy and the possible exercises. With the help of programmers the rules, strategy and the exercises can then be incorporated in `ideas`.

1.3 Student models for rule and strategy based e-tutoring

Rules and strategies describe what a student should do to solve an exercise. Rules and strategies do not describe the behaviour of the students, nor the properties of the rules. Teachers can be interested in the behaviour of the students and the properties of rules. Questions that teachers might have are:

- How difficult are the different rules in the domain?
- How well are the students doing?

With this information a teacher could select suitable exercises for each student. A student may be interested in knowing how well he/she is doing too.

We are interested in describing student behaviour and properties of the rewrite rules ourselves to improve our e-tutoring system. If we can define and implement how a teacher selects suitable exercises, we can improve our e-tutoring system by implementing this selection process. We could also try to improve the e-tutoring system by modelling a students competence, and try to select more appropriate feedback based on competence estimation. For example, we could let the system focus on teaching the separate rules first, and once a student has sufficiently mastered the rules, focus on perfecting the solving strategy. Finally, we could use the information about student

behaviour per rule, to diagnose which rules a student may have problems with.

For improving our e-tutoring system, we will have to create a model which describes, and predicts, the behaviour of students: a student model.

1.4 Research questions and scope

With this research we attempt to create a student model for rule based e-tutoring systems. We will therefore answer the following question: *How can we describe student behaviour in a rule based e-tutoring system with a student model, and estimate the probability of a student applying a rule correctly, the next time he/she tries to apply it?*

Note that the purpose of answering this question is to improve the e-tutoring system. The model should include a student's competence and account for learning. Furthermore we want the model to describe the rules as well as the students. Therefore we include difficulty and discriminativity as rule properties. The operational definition of *difficulty* is: the competence level a student needs to have, for a probalility of succes to equal 0.5. A high *discriminativity* says that the probability that a student with a competence above the difficulty of the exercise, applying the rule correctly is high.

Both difficulty and discriminativity of rules are components in the student model, because they depend on the student population.

In this research project we will limit ourselves to creating and validating the student model. The application of the model in e-tutoring systems is out of scope for this project, and will be subject of future research.

1.5 The result: a student model

The result of this study will be a student model for e-tutoring systems. The central concept in our student model is competence: all other concepts (discriminativity, difficulty and learning speed) are linked to competence. In this research we will make the concept of competence operational, by creating a quantitative measure for it as a parameter of the student model.

Liu [Liu09] states that there are four important aspects to consider while using a competence measure:

- Measurement - this is the most important concept in this study. We create a measure of competence for rule based e-tutoring systems.
- Student population - the group of students to which the measure applies. We use a test student group for obtaining data to test our

measure with. We need to define who these students are in relation to the domain.

- Content - the domain to which the measure applies. Students will be competent on a certain domain. For this study we select one domain to do measurements with.
- Judgement - the setting of a cutoff score to determine whether a student is competent enough or not. We will not do this, as we are only interested in creating the measure. Judgement is thus out of scope for this study.

We create a measure for e-tutoring systems. In order to know whether our measurement is any good we need to define the quality of our model. There are two ways to do this, which we will do both defining a quality measure in terms of the data, and comparing the model with the opinions of domain experts, in order to see how well the model corresponds to reality.

The data we can observe is which rules a student applies, and whether this application was correct or not. Whether the application was correct or not, we call the *outcome*. Outcomes are dichotomously scored, and thus nominal data. From this nominal data, we go to a quantitative measure for competence using the student model we create. The model describes the data by estimating the probability that an outcome will be correct. For new data we can predict the outcome of a student applying a certain rule. When the probability of a correct application is larger than or equal to 0.5 we say that the model predicts a correct application of the rule. The description and prediction of outcomes is thus a *classification* problem. How well the model we create can predict new outcomes, is a measure for the quality of the model. This measure is called the *accuracy of prediction* or just *accuracy*.

Accuracy as a quality measure is however not enough to validate our model. Even though a model may describe the data very well, the model could have no basis in reality: the model could overfit the data it was learned from. We therefore need the opinion of domain experts. The domain experts we interview are teachers in the domain (the content) we perform the measurement with. We therefore select a domain for which these domain experts are available.

1.6 Research project description

In order to answer our research question we perform the activities shown in Figure 1. We start with two paralel activities, selecting a domain and selecting a framework to base the student model on.

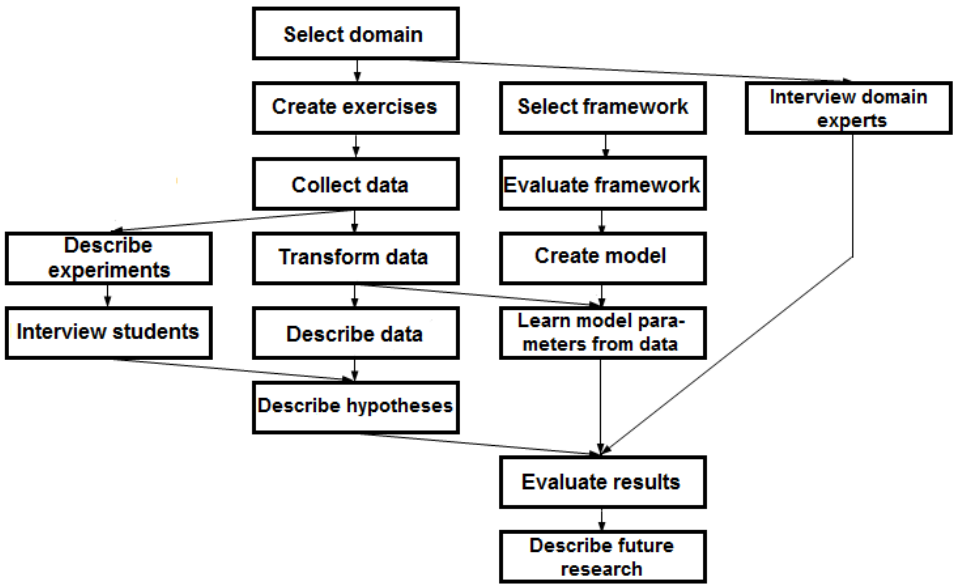


Figure 1: The activity flow of this research project. Each activity is shown in a box. An arrow from activity A to activity B means that A needs to be completed before B can commence.

The domain we select will define the contents of our measurements. We select the domain of formal logic, and more specifically the task of rewriting a logic expression into disjunctive normal form (DNF). We describe the domain and the strategy for the task in chapter 2. We choose this domain and task because:

- It is available in `ideas`. We do not need to define the strategy ourselves, so that we can focus on answering the research question directly.
- It is a domain we teach at our own departments, which means that we can find both students and domain experts.
- It contains more than 20, but less than 50 rewrite rules. There is a reasonable amount of rules to test the model on, but not that much that is is impossible to teach the entire strategy.

The exercises we develop, and the reasons behind the exercises, we describe in chapter 3.

We use an existing framework for measuring competence that has already proven itself. We believe that extending such a framework to create a student model for rule based e-tutoring systems, has better odds of producing a good model, than starting from scratch. The framework we select we evaluate using simulations, in order to see whether this is a reliable model, and how this model behaves. We then create our student model, and evaluate this model using simulations again. We make statements about whether this model will be a reliable model for the data we obtain experimentally. We do this in terms of parameter estimate uncertainty and accuracy of predicting (simulated) outcomes. The selection of the framework and the creation and evaluation of the model, we describe in chapter 4.

We describe the acquisition of the data, the experimental setup and the transformations we performed on the data to prepare it for analysis in chapter 5. In this chapter we include a description of the student population, and the sessions in which we collected the data. The student population consists of students at Utrecht University of Applied Sciences (HU)[1] and students at Utrecht University (UU). The first is a university which offers bachelor programs for higher professional education (HBO[2]), the latter offers bachelor and master programs which prepare for a scientific carreer (WO[3]).

[1] The Dutch name of this university is Hogeschool Utrecht.

[2] HBO is an abbrevation of "Hoger Beroepsonderwijs", which is Dutch for "Higher Professional Education".

[3] WO is an abbrevation of "Wetenschappelijk onderwijs", which is Dutch for "Scientific Education"

We describe the data we acquired, and interview students who participated in the experiments. From this we form hypotheses on what the student model parameter estimates from learning from data should look like. Simultaneously we interview domain experts, whom we do not show the experimental data, and ask them to form hypotheses about the rule properties as well. The data description, the interviews with students, the hypotheses based on these two, and the interviews with domain experts can be found in chapter 6. We will use this information to validate our outcomes with.

We learn the parameters from data, and calculate the accuracy of prediction. We compare the parameter estimates of our model to the opinion of domain experts and the hypotheses we define as a result of interviewing students and inspecting the data. We do this in order to validate our model against reality. We present the results of learning the model parameters from data, and the comparison to the hypotheses and the statements of domain experts, in chapter 7.

We summarize our main conclusions in chapter 8. We include future work, in chapter 9, which will be needed to implement and use the model in real life.

2 The logic domain: disjunctive normal form

We use experiments in which we observe students solving exercises. We select the task of rewriting logic expressions to disjunctive normal form (DNF). We make use of the `ideas` environment, and the GenExas web front for `ideas`. We describe the contents of the measurements in this chapter. We describe the logic domain, and the (preferred) strategy to rewrite logic expressions to DNF.

In the `ideas` environment, problem domains are expressed by means of Haskell data types. The logic domain is defined as the following data types [HJG10]:

$$\begin{aligned} Logic &::= Var \mid T \mid F \mid \neg Logic \mid Logic \wedge Logic \mid \\ &\quad\; Logic \vee Logic \mid Logic \rightarrow Logic \mid Logic \leftrightarrow Logic \end{aligned}$$

$$Var ::= p \mid q \mid r \mid s \mid \ldots$$

Note that we use the standard scientific notation for the logic operators, T for true and F for false.

The rewrite rules for this domain transform an expression of the *Logic* type into another expression of the *Logic* type. Examples of these rules are in Figure 2 [HJG10].

Rewrite rules can be combined using strategy combinators. The basic strategy combinators are given in Table 1. Using a subset of all the available rules in the logic domain, and strategy combinators, we can form a strategy. For completeness we remark that a single rule can also be considered a strategy: the most basic available strategy.

We created a strategy to rewrite any given expression of the *Logic* type to DNF. This strategy thus leads to an answer for any exercise we give to the students participating in the experiments.

The strategy for rewriting to DNF depends on some more strategy combinators than the basic ones. These combinators are defined domain specifically[4]. For example, we need `once` which takes a strategy as its first argument. `once` applies this strategy once to the immediate children of the term in the second argument. We also need is `somewhere`, which uses `once` in its definition, in order to apply its first argument, somewhere where it is applicable in its second argument. `bottomUp` and `topDown` do the same, but with a respective preference for down or up the hierarchy.

[4]They can also be implemented using generic programming techniques.

Basic Rules:

Constants:

AndTrue:	$p \wedge T = p$		AndFalse:	$p \wedge F = F$	
OrTrue:	$p \vee T = T$		OrFalse:	$p \vee F = p$	
NotTrue:	$\neg T = F$		NotFalse:	$\neg F = T$	

Definitions:

ImplDef:	$p \rightarrow q = \neg p \vee q$
EquivDef:	$p \leftrightarrow q = (p \wedge q) \vee (\neg p \wedge \neg q)$

Negations:

NotNot:	$\neg\neg p = p$
DeMorganAnd:	$\neg(p \wedge q) = \neg p \vee \neg q$
DeMorganOr:	$\neg(p \vee q) = \neg p \wedge \neg q$

Distribution:

AndOverOr:	$p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$

Additional Rules:

Tautologies:

ImplTaut:	$p \rightarrow p = T$		OrTaut:	$p \vee \neg p = T$	
EquivTaut:	$p \leftrightarrow p = T$				

Contradictions:

AndContr:	$p \wedge \neg p = F$		EquivContr:	$p \leftrightarrow \neg p = F$	

Figure 2: Transformation rules for logical expressions with names as copied from Heeren et al [HJG10].

Using a sequence that uses the choices from the blocks (constans, definitions, etc.) in Figure 2, the following strategy is defined:

```
dnfStrategy =
    label "Constants" (repeat (topDown constants))
    <*> label "Definitions" (repeat (bottomUp definitions))
    <*> label "Negations" (repeat (topDown negations))
    <*> label "Distribution" (repeat (somewhere distribution))
```

A small, but essential extension to this strategy is to incorporate the idea that whenever a tautology rule or a contradiction rule can be applied, it should be applied immediately. Applying a tautology or contradiction rule will introduce new constants (T or F). Constants should be removed imme-

`<*>`	sequence	a `<*>` b means first a, then b
`<|>`	choice	a `<|>` b means either a, or b
`succeed`	succeed	unit step, always succeeds
`fail`	fail	unit step, always fails
`fix`	recursion	fixed point operator so that the property `fix` $f = f($`fix` $f)$ holds
`many`	zero or many times	`many` $s =$ `fix` $(\lambda x \rightarrow (s$ `<*>` $x)$ `<|> succeed`$)$
`many1`	one or many times	`many1` $s = s$ `<*> many` s
`option`	zero or one times	`option` $s = s$ `<|> succeed`
`not`	succeed iff not applicable	`not` s succeeds only when s is unapplicable
`repeat`	as many as possible	`repeat` $s =$ `many` s `<*> not` s
`repeat1`	,, (at least once)	`repeat1` $s =$ `many1` s `<*> not` s
`try`	try	`try` $s = s$ `<|> not` s
$\triangleright$	preferably	only applies the right part if the left part is unapplicable: $s \triangleright t = s$ `<|>` (`not` s `<*>` t)

Table 1: Rewrite step combinators

diately as well, by using a constants rule. This leads to *dnfStrategy2*.

```
dnfStrategy2 =
  whenever   ( (tautologies <|> contradictions)
             <*> (repeat  (topDown  constants)) )
             dnfStrategy
```

whenever can be implemented using the interleave function [HJ11]. The implementation of *whenever* is omitted.

3 Exercises

The purpose of this chapter is to present what the students had to do during the experimental sessions, and how the experiment was set up. In this chapter we present the exercises we developed for our experiments. After presenting these,we show which rules show up in the solutions to these exercises. To also get a visual impresson of the experiments, you can find a screenshot of the GenExas interface for `ideas` in appendix A.

Each student has to solve 25 exercises.[5] For each exercise, the instruction is to rewrite the given expression into disjunctive normal form. The solution generated by the strategy `dnfStrategy2` is no longer than ten steps. We developed the exercises such that the average number of steps necessary to solve the exercise is in between 5 and 6 steps.

The rules that `ideas` accepts, as shown in Figure 4, have to occur more than once in the series of exercises in order to be able to test the hypothesis. We show the rules in (hierarchical) groups of similar rules. During the development of the exercises, the rule groups on level 2 should occur at least once every five exercises, and each rule should occur at least five times. We do this in order to spread out the rules over the series of exercises, and to make each rule frequent enough to say something about the development of the competence for that rule for each student over time. We do not fulfill this requirement for the rules `orOverAnd` and commutativity, as these steps are not enforced steps. Furthermore there are two more tautology rules which are not in this picture ($p \to p$ and $F \to p$). These two rules are not in the picture because they were not in the instruction. They can however be applied by the students.

The excersises that we developed are given in the following list:

✓ $p \to q$ (Example exercise)

1. $\neg(\neg q \wedge \neg r) \wedge r$
2. $((\neg(q \wedge q) \to p) \wedge r) \vee (s \wedge \neg T)$
3. $p \to (\neg\neg q \vee (r \wedge p \wedge \neg p))$
4. $(q \leftrightarrow (r \vee r)) \wedge (s \vee \neg s)$
5. $F \to (p \vee q \vee \neg T)$
6. $p \vee p \vee (\neg(q \vee p) \to r)$

[5] 26 exercises if the example exercise is also counted.

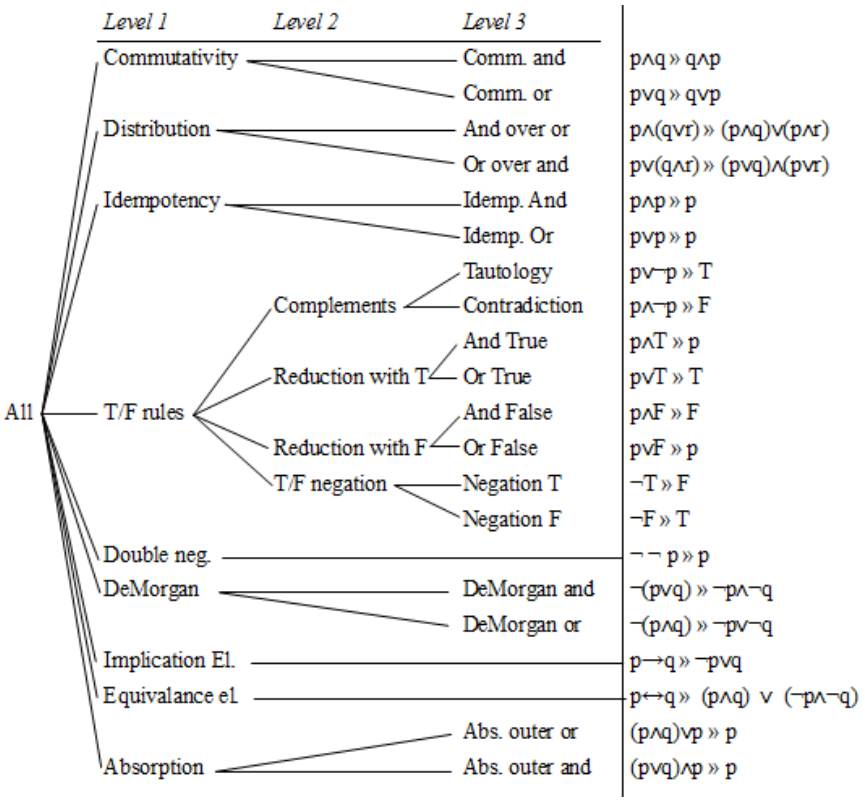


Figure 3: The rules which can be used for solving the dnf exercises, categorized.

7. $(q \leftrightarrow (q \rightarrow p))$
8. $(F \vee \neg\neg\neg(q \wedge r)) \wedge \neg q$
9. $\neg(p \wedge (q \wedge r)) \vee (r \wedge T)$
10. $p \wedge (p \rightarrow (q \vee r))$

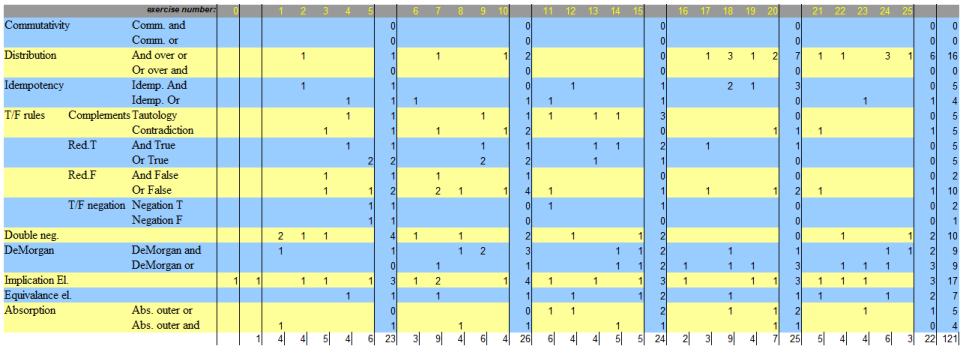
11. $\neg(p \rightarrow p) \vee (r \wedge p) \vee r \vee r$
12. $(((p \wedge q) \vee p) \wedge p) \leftrightarrow \neg\neg q$
13. $(q \vee (q \rightarrow p)) \wedge (r \vee s)$
14. $\neg(p \wedge (q \vee r)) \wedge (((s \vee p) \wedge p) \vee \neg p)$
15. $\neg(p \wedge (q \rightarrow r)) \vee (q \leftrightarrow r)$

16. $((p \vee q) \rightarrow (q \vee r \vee r)) \vee (s \wedge \neg T)$
17. $(p \wedge T) \wedge (F \vee q \vee r)$
18. $(p \vee s \vee (q \wedge r)) \leftrightarrow s$
19. $(s \wedge s) \wedge ((p \vee q) \rightarrow r)$
20. $r \wedge (r \rightarrow q) \wedge (q \vee p)$

21. $(r \leftrightarrow p) \vee (p \wedge (p \rightarrow q))$
22. $(r \vee s) \wedge \neg(q \rightarrow p)$
23. $\neg(q \rightarrow r) \vee \neg r \vee \neg(s \vee s)$
24. $r \leftrightarrow ((s \wedge q) \vee p)$
25. $((\neg p \vee s) \wedge \neg(q \vee \neg r)) \vee \neg s$

`ideas` contains the strategy `dnfStrategy2`. We let `ideas` apply this strategy on the exercises for us, and count how often a student should apply the rules if the student follows this strategy. `dnfStrategy2` is the strategy we want to teach. The rule counts are shown in Figure 4. We see that there are 121 steps to solve all exercises. However, many students will probably use more rules, as they may take detours and follow less optimal strategies. It is also possible that a student avoids some rules, or that the student gives up on an exercise before reaching the point where a rule can be applied.

Note that there are rules that do not appear in strategy `dnfStrategy2`. Examples of such rules are distribution of $\vee$ over $\wedge$, and the commutativity rules. It is impossibile to determine all possible intermediate solutions the



			exercise number:	0		1	2	3	4	5		6	7	8	9	10		11	12	13	14	15		16	17	18	19	20		21	22	23	24	25		
Commutativity		Comm. and									0						0						0						0						0	0
		Comm. or									0						0						0						0						0	0
Distribution		And over or				1					1		1			1	2						0		1	3	1	2	7	1	1		3	1	6	16
		Or over and									0						0						0						0						0	0
Idempotency		Idemp. And				1					1						0		1				1			2	1		3						0	5
		Idemp. Or							1		1	1					1	1					1						0			1			1	4
T/F rules	Complements	Tautology							1		1				1		1	1		1	1		3						0						0	5
		Contradiction						1			1		1			1	2						0					1	1	1					1	5
	Red.T	And True							1		1				1		1			1	1		2		1				1						0	5
		Or True								2	2				2		2			1			1						0						0	5
	Red.F	And False						1			1		1				1						0						0						0	2
		Or False						1		1	2		2	1		1	4	1					1		1			1	2	1					1	10
	T/F negation	Negation T								1	1						0	1					1						0						0	2
		Negation F								1	1						0						0						0						0	1
Double neg.						2	1	1			4	1		1			2		1			1	2						0		1			1	2	10
DeMorgan		DeMorgan and				1					1			1	2		3				1	1	2			1			1				1	1	2	9
		DeMorgan or									0		1				1				1	1	2	1		1	1		3		1	1	1		3	9
Implication El.				1	1			1	1		3	1	2			1	4	1		1		1	3	1			1	1	3	1	1	1			3	17
Equivalance el.									1		1		1				1		1			1	2			1			1	1			1		2	7
Absorption		Abs. outer or									0						0	1	1				2			1		1	2			1			1	5
		Abs. outer and				1					1			1			1					1	1					1	1						0	4
					1	4	4	5	4	6	23	3	9	4	6	4	26	6	4	4	5	5	24	2	3	9	4	7	25	5	4	4	6	3	22	121

Figure 4: Per exercise we counted the rules that would be applied according to `ideas` when the `Worked-out exercise` button is pressed at the start of the exercise. We took effort to select exercises which balance the number of occurences of the rules. Some rules probably have too few occurences to be able to say much about the competence of the students for these rules. Some rules occur more often than others: implication elimination (17), distribution of $\wedge$ over $\vee$ (16) and double negation, and $\vee$ false (10) are the rules which occure most often.

student will get, and the frequencies with which they will arise during the experiment beforehand. A student choses his/her own solution path, which may contain clever tricks.A student sometimes applies rules without a plan, just to see whether they will see a solution in the situation that then arises. There are also rules that have not been explained, and are not foreseen in the strategy, but can be applied by students (they either know of the rule, or infer the rule by logical reasoning).

4 Creating the student model

4.1 How to create a student model?

In this chapter we will create a student model including competence, learning, difficulty and discriminativity. The last two properties are concepts that describe rules. The properties of the rules are relative to the student population. Therefore, both rule and student properties belong in the student model.

The model will make the concepts of competence, learning, difficulty and discriminativity operational. We do this by linking these concepts through mathematical formulae to the outcome of the application of a rule by a student.

First let us look once more at the intuition behind the four concepts. Competence and difficulty are two concepts closely related. More difficult rules can be performed well by more competent people, while less competent people can only handle less difficult rules. While making the concepts of difficulty and competence operational, the measures for difficulty and competence should thus be on the same scale. Discriminativity means how well a rule can distinguish between students, and how certain it is that a student with a high competence relative to the difficulty of a rule, will perform that rule correctly. Learning is the increase of competence with each attempt at the application of a rule, and the feedback provided because of it.

The student model we create includes these concepts. Difficulty and discriminativity are properties of rules, and (start) competence and learning speed are properties of students.

We define the quality of the student model in terms of the data, and in terms of reliability of parameter estimates. The most important quality measure is how well the model predicts new data: the *accuracy*. The other important quality is the model's reliability: if we generate data using the model with a certain instantiation of the model parameters, how well can we learn these parameters back from data? The reliability tells us how well we should trust our parameter estimates. The accuracy tells us how well the model relates to the data.

When we create a student model, it should include competence, learning, difficulty and discriminativity. We try to accomplish a model with these qualities.

We will use an existing theory for the creation of our student model. We did not find a theory that could be applied directly to rule based e-tutoring systems. We therefore use a theory that includes many of the concepts we

want in our student model already, and extend it. We extend an existing theory rather than starting from scratch, because we expect that this will lead to a higher quality and will correspond well to reality. Also, we hope that our model will be easy to understand when already familiar with the existing theory.

4.2 Item response theory

As a foundation for creating our student model for rule based e-tutoring systems we use Item Response Theory (IRT). IRT includes an operationalisation of competence, difficulty and discriminativity. IRT is a well-established theory, which has proven itself in the field of educational measurement.

IRT is a theory that has its roots in the field of psychology. While being a well-established theory of which the roots can be traced back to the beginning of the twentieth century [Bak85], IRT is an active field of research. In the introduction to the second edition of their book, Baker and Kim [BK04] describe IRT as a "dynamic field of inquiry", and give various examples of recent articles. IRT has proven to be useful in the field of education and psychology for measuring and analysing the results of tests [Lor80]. IRT has been applied to computer science eduction in particular too. Sudol and Studer [SS10], have recently presented an article on a symposium about computer science education about two models included in IRT. They use the R statistical package to analyse exams using item characteristic curves.

IRT revolves around these item characteristic curves. An item characteristic curve is the function which relates the probability of a correct response to the competence of a student for one item [Bak85]. The curve relates competence to the probability of a correct response (outcome), using difficulty and discriminativity. The term *item* is used for that what a person reponds to, which leads to an outcome. In our case an item is a rule.

In IRT several versions of the item characteristic curve exist, which we will discuss briefly.

4.3 The different models within IRT

The most important class of item characteristic curves included in IRT are the logistic ogive functions.[6] The logistic ogive functions use instances of the

[6]The other class are the normal ogive functions. These are, accoring to Baker and Kim [BK04], not used in practice and will therefore not be discussed any further here. We could not find a reason for this, other than that the parameters are much more difficult to estimate, and that the other models are easy to interpret.

logistic sigmoid function which is used in the field of machine learning for classification and probability estimation (see Bishop [Bis06], section 4.3.2). The greek letter σ is used as a symbol for the logistic sigmoid function:

$$\sigma(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}} \tag{1}$$

Note that $\sigma(x)$ has values between 0 and 1, and can thus be used as a probability function. We can substitute any function or combination of functions for x. In logistic regression x is a linear combination of functions. The logistive ogive functions used in IRT use instances of the logistic sigmoid function, with different substitions for x.

We will describe the different substitutions used in IRT from basic to complex and chose one of these as the foundation for our own student model.

The one parameter logistic ogive model (1PL), or Rasch model, is described by $P(o_i = 1) = \sigma(\theta - b_i)$. $P(o_i = 1)$ is the probability that item i will be answered correctly. θ is the competence of a student. b_i is the difficulty of item i. The difficulty of an item, and the competence of a student are on the same scale. Both b_i and θ range from negative infinity to positive infinity.

The two parameter model (2PL) introduces the discriminativity of an item a_i, in $P(o_i = 1) = \sigma(a_i \cdot (\theta - b_i))$. The range of b_i and θ stay the same. a_i ranges from negative infinity to positive infinity too.

The three parameter model (3PL) also includes a guessing probability for each item: c_i - the probability that a student with the lowest possible competence will provide the right answer.[7] The function we then obtain is: $P(o_i = 1) = c_i + (1 - c_i) \cdot \sigma(a_i \cdot (\theta - b_i))$. This function is a linear combination that includes the logistic sigmoid function. It is no longer a logistic sigmoid function itself.

Of the three available functions we choose 2PL. 2PL makes difficulty, discriminativity and competence operational. Furthermore we do not need a guessing probability. We assume that the effect of guessing is negligible. This assumption is justified by the fact that the students will have to input a new intermediate solution by themselves each step, and cannot pick an alternative out of a small list.

[7] $\lim_{\theta \to -\infty} P(o_i = 1) = c_i$

4.4 The 2PL function

In the two parameter logisitic ogive function, the probability of student i answering an item j correctly is:

$$P(o_{i,j} = 1|\theta_i, b_j, a_j) = \sigma(a_j(\theta_i - b_j)) = \frac{e^{a_j(\theta_i - b_j)}}{1 + e^{a_j(\theta_i - b_j)}} \quad (2)$$

The outcomes $o_{i,j}$ are dichotomously scored. An outcome is either 1, which means correct, or 0 which means incorrect. $P(o_{i,j} = 1|\theta_i, b_j, a_j)$ is thus the probability of a correct answer for item j by student i.

Competence is made operational by defining the competence parameter θ. Note that this is measure that cannot be directly observed. It has no natural 0 which can be directly measured (like 0 Kelvin for temperature). Therefore Baker [Bak85] introduces the measurement of competence with an arbitrary zero (as the mean). The values of θ usually range between -3 and 3 in educational measurement [Bak85].

The difficulty is made operational in terms of competence. If an item has a difficulty equal to the competence of a student answering to the item, the probability of a correct outcome is 0.5. In theory item difficulties range from negative infinity to positive infinity, but within educational measurement usually range between -3 and 3 as well [Bak85].

Discriminativity can be interpreted as follows: there is a higher probability to make an error even though a student knows the rule (high competence) if the discriminativity is low. When the discriminativity is high you either master the rule or you do not, and once you have mastered it there is a very small chance of still making an error with it. This explanation we also give to the domain experts when defining discriminativity.

2PL makes discriminativity operational using the parameter a_j.

When we look at graphs for the 2PL function, which are shown in Figures 5 and 6, we can see that the operationalisation of competence, difficulty and discriminativity corresponds to the intuitive description of these concepts. Note that when b_j increases the curve moves to the right, and when a_j increases, the curve becomes steeper. For a student with a competence θ the probability of a correct outcome is 0.5 when $\theta = b_j$. If $\theta > b_j$, the probability of a correct outcome increases when a_j increases. If $\theta < b_j$, the probability of a correct outcome decreases when a_j increases.

IRT, and more specifically the 2PL function, cannot be applied to rule based e-tutoring systems without changes. This is because the 2PL model does not contain any time-depence: the competence of a student is assumed to be constant. This assumption is okay for tests, but of course not for

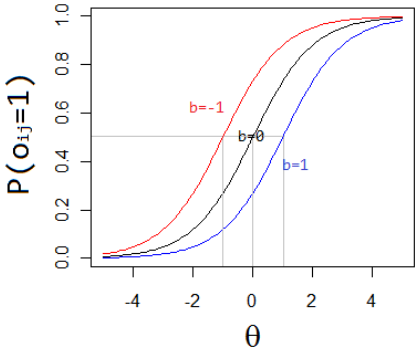


Figure 5: IRT curves varying b

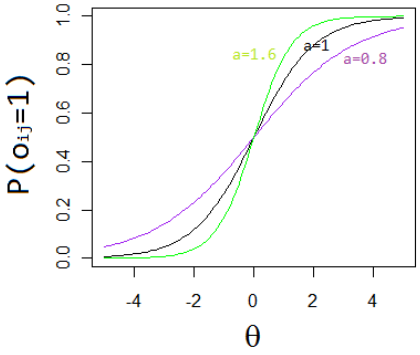


Figure 6: IRT curves varying a

e-tutoring systems. Furthermore IRT assumes one outcome per item. We have many outcomes for the same rule in e-tutoring systems.

4.5 The roadmap to creating a student model for rule based e-tutoring

We create a model for strategy and rule based e-tutoring systems. We do this using the 2PL function as a starting point. This function is lacking a learning parameter, which we add.

We want to create a student model along with learning algorithms to learn the parameters of the model from data. We have to consider that the amount of data we gather is small. The amount of available data can have a large impact on both the algorithms and the results for parameter estimation, as we will show.

As the amount of data is small we will have to check the behaviour of the algorithms for parameter learning for the model for small amounts of data. For this purpose we will use parameter recovery simulations. We will have to adapt the standard algorithms available for 2PL if we run into problems due to the amount of data.

We use the learning algorithms of the parameters in the 2PL model described in literature. We will make use of the algorithms described by Baker and Kim [BK04], though we will explain the technique of iteratively reweighted least squares following Bishop [Bis06]. Bishop uses a shorter notation and a more general approach. We will adapt the algorithms where necessary, to be able to handle small amounts of data. While implementing the parameter estimation techniques, we will continuously look at how the

learning algorithms behave.

The steps we will take to create our student model for rule based e-tutoring systems are:

1. First we implement parameter learning, prediction, and simulation. For this we follow literature [BK04][Bis06] where we can. Where we need to change the algorithms to be able to handle small amounts of data, we will mention this. An introduction to 2PL parameter learning and the objectives of which are in section 4.6.

    - We implement the learning of the item parameters in 2PL when the student parameters are known. We show that the parameters are identified - meaning there are a unique parameter estimates that minimizes the likelihood of the data. We present how to learn the item parameters in section 4.7. We include simulations to show how the learning behaves with different amounts of data in section 4.8.
    - We implement the learning of the student parameters in 2PL when the item parameters are known. These parameters are again identified. We present the description of the implementation in section 4.9. We use simulations to see how the learning behaves with different amounts of data. We present the results of these simulations in section 4.10.
    - We implement simultaneous learning of the item and student parameters in 2PL. We follow the EM-like learning procedure described by Baker and Kim [BK04]. We will point out that this model as a whole, is no longer identified. Because we are interested in the order of competence, difficulty and discriminativity this need not be a problem. We show the standard implementation for simultaneous parameter learning, following literature, in section 4.11. In this section we also show that what we are interested in - the order of the parameter estimates, is preserved through parameter recovery, even though the model is not identified. We illustrate this by a parameter recovery simulation.
    - When we try small amounts of data we run into problems. The standard algorithms do no longer converge. We thus create a discrete version of simultaneous parameter estimation, which always converges in section 4.12. We show by simulations that this algorithm produces similar results to the standard algorithms.

We show by parameter recovery simulations how this algorithm behaves for small amounts of data.

- In section 4.13 we look at using the Rasch model (1PL), as an alternative to 2PL, as suggested in literature.
- We look at how well the 2PL model predicts simulated data in section 4.14. Prediction of outcomes for new data is done by continuously updating the student parameters. We look at the estimates for competence and probability. Finally we look at the accuracy of predicting new outcomes with the 2PL model.

2. We summarize what we have learned from the 2PL function and look at which lessons can be learned for creating a student model for rule based e-tutoring systems in section 4.15.

3. In section 4.16 we look at several possibilities to extend the 2PL function to a student model for rule based e-tutoring systems. We evaluate the chosen model using simulations in terms of the reliability of the parameter estimates, for large and small amounts of data.

4. We will look at how well the model predicts simulated data in section 4.17. The results of this simulation can directly be compared to the results we obtain from applying the model to real data we obtain from experiments.

5. Finally we will summarize the model we created and our findings from simulations in section 4.18.

Note that in the end, our model will not be identified - there is no unique combination of parameter values that optimize the loglikelihood. However, we are only interested in the accuracy of the model, which is determined by the probabilities which can be calculated with the parameters, and in the order of the parameter estimates for each parameter. We show that in this respect, a unique set of parameter values is not a requirement, as long as the order of the parameter values is preserved.

4.6 Learning the item and student parameters

As it is a probabilistic model, the parameters of the 2 parameter logistic ogive function (2PL) can be learned from data. The data is in the form of an n by m matrix with binary values, where n is the number of items, and m is the number of students. 0 denotes a wrong answer and 1 denotes a

correct answer. The item parameters can be used to validate tests/exercises, or just as a description of the items. The student parameters can be used to describe the student group. We could also attempt to predict outcomes (probability estimation) for new students who have completed a subset of the items.

First we observe that if the values of either the student parameter θ_j or the item parameters a_i and b_i are known, the other parameters can be learned easily by maximum likelihood estimation (MLE). In the case of learning θ_j with known a_i and b_j, all student results are independent, and the likelihood can be optimized by seperately optimizing all θ_j. The same holds for known θ_j, which makes all the items independent. However, usually none of the parameters is known.

We follow the learning procedure for 2PL as described by Baker and Kim [BK04]. The learning algorithms we later adapt to handle our extension to IRT - our student model.

Note that our interest in item and student parameters is in the order of the parameter estimates. We want to know which items are more or less difficult/discriminative than others. We want to know which students are more competent than others. The order of the parameter estimates is the only thing we can compare to the opinion of domain experts.

By following a known learning procedure and doing parameter recovery simulations, we implement learning algorithms we can reuse for our own student model. We also see how the learning of 2PL parameter values behaves for smaller datasets. We evaluate what we have to pay attention to when creating our student model for e-tutoring systems.

4.7 Item parameter (a_i and b_i) learning

Learning the item parameters (a_i and b_i) per item, in a 2PL model is a subproblem of learning all parameters in this model simulataneously.

The subproblem of finding optimal parameter values for a_i and b_i for known competence θ_j of all students, is an important subproblem. For each item individually the values for a_i and b_i can be determined. This is because for known θ, the outcomes for every item are independent of the outcomes of all other items. The data per item is m outcomes, combined with m values of θ.

Optimizing the likelihood of a data set assumed to be described by 2PL IRT with respect to a_i and b_i with known θ_j can be done by logistic regression, using iteratively reweighted least squares (IRLS). A global optimum can be approximated by IRLS. The error function is a concave function

in the parameters that need to be estimated and therefore has a unique minimum (if it has any) [Bis06].[8]

IRLS is an iterative algorithm for minimizing an error function E, with respect to the parameters that need to be estimated. Each iteration, the new parameter values $\mathbf{w}_{new}$ are calculated using the previous estimates using the update formula in equation 3. In this equation H is the Hessian matrix (matrix of the second derivatives of the error function). $\nabla E(\mathbf{w}_{old})$ is the gradient of the error function, evaluated in the current weights. When we form the problem such that the weights are the parameters we want to estimate, we can use this procedure to optimize the weights with respect to the loglikelihood.

$$\mathbf{w}_{new} = \mathbf{w}_{old} - H^{-1}\nabla E(\mathbf{w}_{old}) \tag{3}$$

Using the cross entropy function[9] for E in procedure (3), we obtain the formula in equation 4, where the parameters to be estimated are in $\mathbf{w}$ in $\sigma(\mathbf{w}^T\phi)$.[Bis06]

$$\mathbf{w}_{new} = \mathbf{w}_{old} - (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T(\mathbf{y} - \mathbf{t}). \tag{4}$$

In this equation $\mathbf{t}$ is the target vector, which contains the outcomes for all the students of an item, $\mathbf{y}$ are the probabilities of a positive outcome predicted with the current weights, and $\mathbf{R}$ is a diagonal matrix where the elements are defined as $R_{nn} = y_n(1 - y_n)$. Φ is the input matrix, for which each row represents the evaluations of the input functions θ and 1 for one data point.[10]

In the case of recovering a and b, for given θ the corresponding logistic function can be written as $\sigma((a, -ab)\cdot(\theta, 1))$[11]. The weights are $\mathbf{w} = (a, -ab)$ and Φ ise a matrix with as many rows as students, and with $(\theta_j, 1)$ as each row. Note that θ_j is different for each student, and 1 is a constant.

We implemented logistic regression for item parameter learning in JAVA.[12]

[8] Our description is equivalent to that of Baker and Kim [BK04]. We chose this description as it is more compact, and more common in the field of machine learning. It is also more general, as it is written as an instance of standard logistic regression.

[9] The negative loglikelihood

[10] Generally speaking, the input functions are known functions of the existing data. Any set of functions can be substituted, as long as they can be evaluated with known data. The evaluated values of the input functions form the rows of the input matrix Φ.

[11] Note the vector product here. $(a, -ab) \cdot (\theta, 1) = a\theta - ab = a(\theta - b)$

[12] The code of this implementation will be made available on request. Please e-mail the author at `diederik.roijers@hu.nl`.

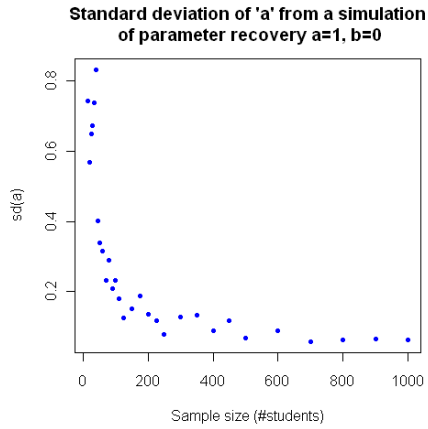


Figure 7: The internal standard deviation in a in a sample of 15 as a function of the number of students in item parameter recovery of $a = 1$ and $b = 0$.

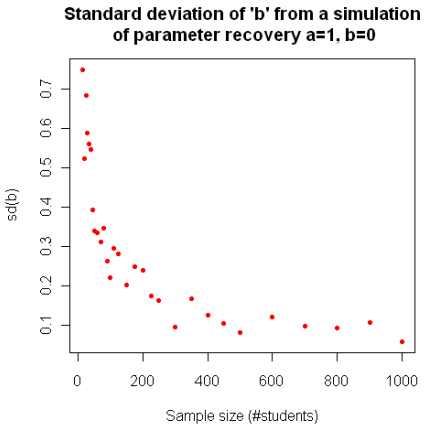


Figure 8: The internal standard deviation in b in a sample of 15 as a function of the number of students in item parameter recovery of $a = 1$ and $b = 0$.

4.8 Item parameter (a_i and b_i) recovery simulations

Using the implementation of logistic regression for item parameter learning in JAVA we have performed item parameter recovery simulation studies, with varying the numbers of students.

In the first simulation the 'true' parameters to be recovered were $a = 1$ and $b = 0$. θ was uniformly distributed between -3 and 3. Each simulation was repeated 15 times, for the same θs, and the internal standard deviation within those groups of 15 estimates was calculated. These standard deviations are shown in Figures 7 and 8. The standard deviations seem to go down roughly exponentionally with the number of students increasing.

From the above examples it follows that the standard deviation decreases with the number of students. This is also the case for different true b's. In the case that the mean of the student competences, which is in the same dimension as b, is far away from the true value of b, the standard deviation goes quickly up again. In Figure 9 this is shown for various numbers of students. For these measuraments θ was uniformly distributed between -3 and 3. For all student population sizes b explodes when the true value of b is close to or smaller than -3 or close to or greater than 3. Note that the standard deviation also decreases with the number of students, for larger

and smaller values of b.

The explosive character of the standard deviation becomes even clearer when the true value of b is outside of the competence spread. As can be seen in Figure 10, both the standard deviations in the estimates of a and b blow up. This blowing up goes on to the extent that $\sigma(\hat{b}) \approx 0.8$ at $b = \pm 4$. At that high standard deviations, any conclusions about the item parameters are dubious, even at student population sizes of 300.

Simulations using small (simulated) population sizes for values of b that differ a lot from the range of θ are extremely difficult, due to the high risk of linear separability or even only outcomes of one kind (positive or negative). In the simulations, samples with a size less than 150 outcomes proved to be undoable. Linear separability occurs when there exists a θ_s such that for all students with a competence smaller than θ_s all outcomes are 0 and for students with a competence larger than θ_s all outcomes are 1. The probability of this happening increases with small amounts of data, and when the values of θ are far away from true b. This is because the probability of a correct answer with $b >> \theta$ is very small, as can be deduced from the formula in equation 2.

4.9 Student parameter (θ_j) learning

When the item parameters are correctly estimated, the next important step is to estimate the competences for students performing the series of items. In this case it is not possible to write the IRT function in a way that standard logistic regression can be applied to learn a weight θ. This is because $a(\theta - b)$ is not a linear combination of functions on the known input data (a and b) and thus the probability function cannot be written as $\sigma(\mathbf{w}^T \phi)$. Equation 3 is written for this special case. First it is rewritten to the 1-dimensional[13] form in equation 5.

$$\theta_{new} = \theta_{old} - \frac{d^2 E(\theta_{old})}{d\theta_{old}} \frac{dE(\theta_{old})}{d\theta_{old}} \tag{5}$$

For E we use the cross entropy function for IRT (equation 6), which is the negative logilikelihood [14]:

$$E(\theta) = -\sum_{n=1}^{N} \left(t_n \ln(\sigma(a_n(\theta - b_n))) + (1 - t_n) \ln(1 - \sigma(a_n(\theta - b_n)))\right) \tag{6}$$

[13] There is only one weight: θ_j.

[14] Minimizing the cross entropy thus means maximizing the (log)likelihood.

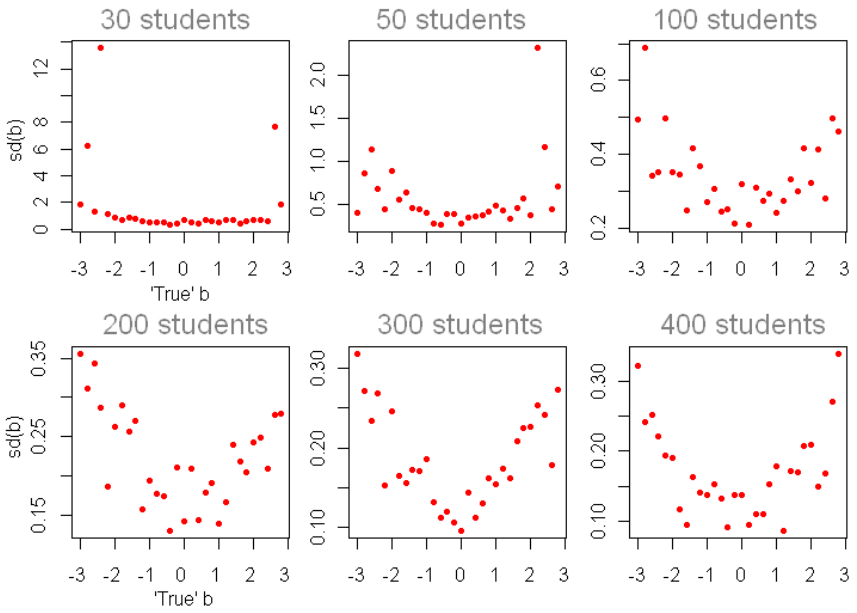


Figure 9: The internal standard deviation in estimated b in a sample of 15 as a function of the true value of b, in item parameter recovery simulation with $a = 1$ and θ uniformly distributed between -3 and 3. Note that the y-axes, denoting the standard deviation in the estimated value of b, is a different scale for each diagram, and the overall standard deviation is going down as the number of simulated students increases.

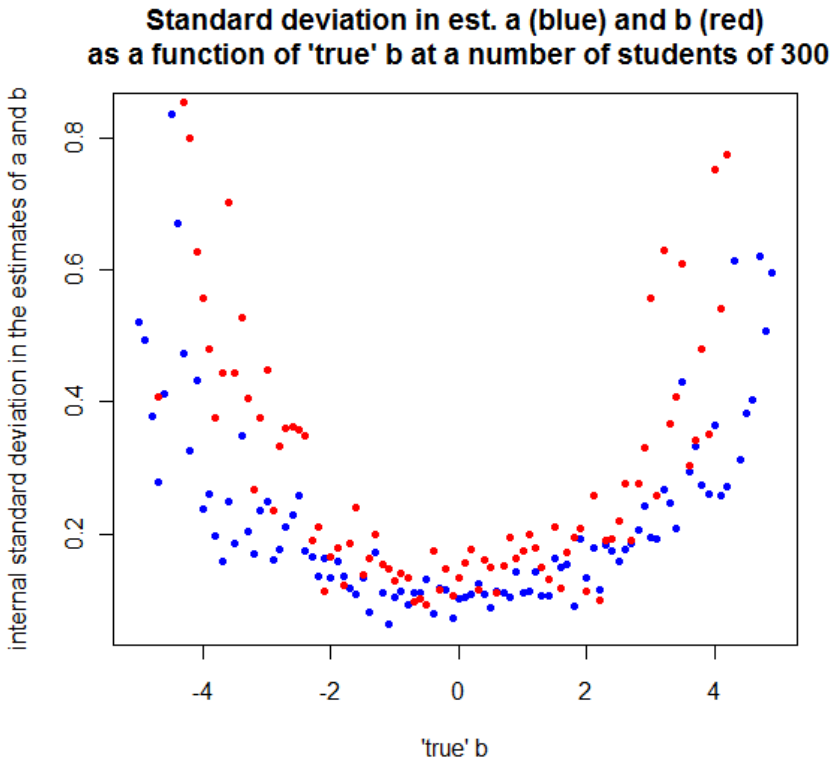


Figure 10: The internal standard deviation in estimated a and b in a sample of 15 as a function of the true value of b, in item parameter recovery simulation with $a = 1$ and θ uniformly distributed between -3 and 3. Note that the red dots denote the standard deviations in $\hat{b}$ and the blue dots the standard deviations in $\hat{a}$.

where N is the total number of items.

Combining equations 5 and 6, the IRLS update function for IRT in θ is obtained (equation 7). Note that the Hessian matrix (in this case only the second derative of the cross entropy function) is positive definite, as $0 < \sigma(a_n(\theta - b_n)) < 1$, which means that the error function is a concave function in θ and thus has a unique minimum.

$$\theta_{new} = \theta_{old} - \left[\sum_{n=1}^{N} a_n^2 \sigma(a_n(\theta_{old} - b_n))(1 - \sigma(a_n(\theta_{old} - b_n)))\right]^{-1} \cdot \left[\sum_{n=1}^{N} a_n(\sigma(a_n(\theta_{old} - b_n)) - t_n)\right] \tag{7}$$

IRLS for estimating a single student parameter is implemented in JAVA, using a convergence stop criterium (when $\|\theta_{new} - \theta_{old}\| < 0.0001$ the iteration is stopped). Running the JAVA implementation of formula 7, we observe two problems: the program does not terminate for start values $\theta_{j,t=0}$ that are far away from the true value of θ_j. The program sometimes does not terminate due to oscillation between two values, and thus not converging. Here, the problem is typically caused by a small difference between the two values, just above (about a factor 10 above) the stop criterion.

The first problem is prevented by first trying out different start values for $\theta_{j,t=0}$, and calculating the loglikelihood. The iterative update procedure is then started using the start value with the highest likelihood. The true values of θ are always chosen to be between -3 and 3. Trying out the start values is done in three steps. First the values ranging from -10 to 10 are tried, in 20 steps. The θ value with the lowest loglikelihood is calculated and stored (let us call it $\hat{\theta}_1$). Then the values between $\hat{\theta}_1 - 3$ and $\hat{\theta}_1 + 3$ are tried out in 20 steps, leading to $\hat{\theta}_2$. In the interval $\hat{\theta}_2 - 1$ and $\hat{\theta}_2 + 1$, again 20 steps are tried. The value that yields the lowest loglikelihood of that last step is taken as the start value for the iterative approach: $\theta_{j,t=0}$.

The latter problem is tackled by using a heuristic. After 100 iterations, the algorithm is stopped. The values of θ_j after each iteration are kept in a list. This list is used to find any values of θ_j that appear twice or more. If a value appears twice, this indicates oscillation. From the list of all θ_j estimates after each iteration, we extract the θ_j values that are in the list twice. These values are the ones θ_j oscillates between. We find the maximum value of these θ_j values which the algorithm oscillates between, and the minimum value. We calculate the loglikelihood of 100 values between these two values. The 100 values are chosen on regular intervals. The value that

yields the smallest loglikelihood we use to restart IRLS with. This resulted in immediate convergence in all instances we tried this.

4.10 Student parameter recovery simulations

With the above implementation we perform parameter recovery simulations. In the simulations for estimating θ, the item parameters are randomly chosen per simulation. Each item paramter b is drawn from a uniform distribution between -3 and 3, and (idenpendently) item parameter is drawn a from a uniform distribution between 0.8 and 1.8[15]. Using the same set of item parameters, data is randomly generated on the basis of a true θ, 15 times. Fifteen θ estimates are made. The mean and internal standard deviation of those 15 estimates are calculated.

For the same true θ, it has been shown that the standard deviation goes down, when the number of items used to estimate θ goes up. Figure 11 shows that this decrease seems to be exponential in the sample size. Figure 12 clearly shows that estimates of θ are bad when the 'true' θ is outside the range of the spread of b (and vice versa). Caution with accepting an estimate for either b or θ of ≤ -2 and ≥ 2 is required.

Also for the student parameter (θ) the standard deviation "explodes" when the values of θ are outside the range of b. Closer to the edges of the range of b the standard deviation in estimated θ (Figure 13) does not blow up as quickly as the standard deviations for estimated b (Figure 9).

4.11 Simultaneous student and item parameter learning

For simultaneous recovery of student and item parameters we use a simple iterative EM-like approach, following Baker and Kim [BK04]. First, initial estimates are set for a_i and b_i. Then we estimate the student parameter, after which, we estimate the item parameters with the new values for the student parameter. With the new estimates for the item parameters new estimates are made for the student parameter, and so on.

For the initial values of a_i we use 1.0, which corresponds to the Rasch model. We do this because no prior information exists about the items, so we cannot say that one item is more discriminative than another. For b we make a rough estimate, based on the outcomes (the dichotomously scored simulated item responses). For each item i the number of correct answers, cnt_i, is counted. Let us call the maximum number of correct answers cnt_{max}

[15]This range is similar to that which Hulin et al [HLD82] use, though they use a complexer distribution. We used a uniform distribution for the sake of simplicity.

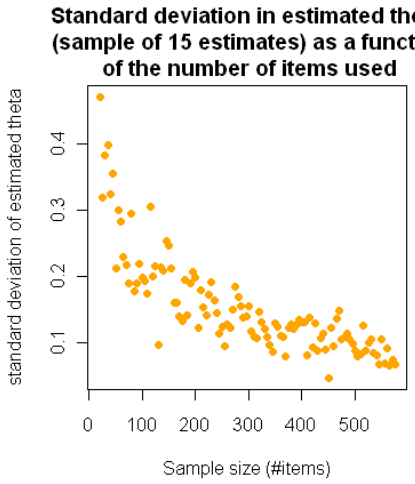


Figure 11: The internal standard deviation in estimated $\hat{\theta}$ in a sample of 15 as a function of the number of items used to estimate the student parameter, in a student parameter recovery simulation with 'true' $\theta = 1$ and a and b uniformly distributed between 0.8 and 1.8 and between -3 and 3. Note that the orange dots denote the standard deviations in $\hat{\theta}$.

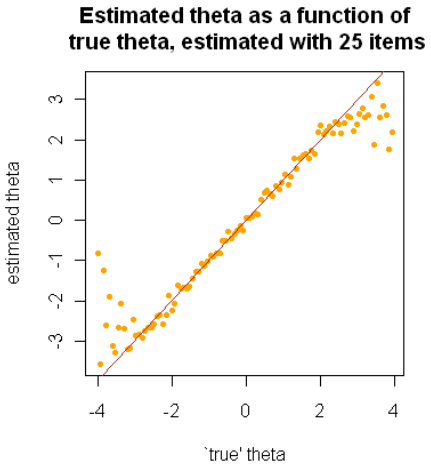


Figure 12: The estimated value of θ as a function of the true value of θ. Estimated in a simulation with 25 items a and b uniformly distributed between 0.8 and 1.8 and between -3 and 3.

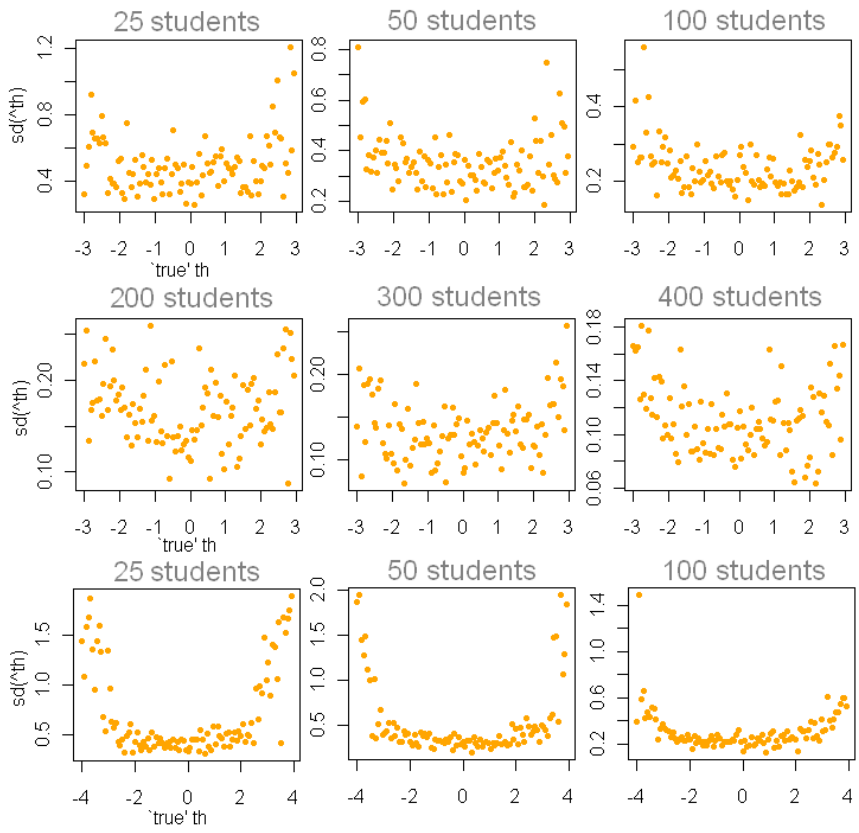


Figure 13: The internal standard deviation in estimated $\hat{\theta}$ in a sample of 15 as a function of the true value of θ, in a student parameter recovery simulation with a and b uniformly distributed between 0.8 and 1.8 and between -3 and 3. Note that the y-axes, denoting the standard deviation in the estimated value of $\hat{\theta}$, is a different scale for each diagram, and the overall standard deviation is going down as the number of simulated students increases.

and the minimum number of correct answers cnt_{min}. The initial estimate for b_i is $\frac{6 \cdot (cnt_i - cnt_{min})}{cnt_{max} - cnt_{min}} + 3$. This results in a minimum value of b_i of -3, and a maximum value of b_i of $+3$. This is a heuristic to put the difficulties of the items in more or less the right order, on the scale -3 to 3 which is recommended by Baker and Kim [BK04].[16]

The EM-like algorithm using IRLS for both item and student parameter estimation as building blocks, converges for samples with more than 150 students and 150 items in all the simulations we performed. We observe (Figure 14), in a 250 students and 250 items simulation, that for the heuristically determined initial values of b, the algorithm convergences in a typical way: after a very steep drop, convergence is exponential. We also observe (Figure 15), that although the 'true' value of b and the estimated value of b are highly correlated, the slope of the estimated values when plotted against the true values is slightly steeper than 1, in this simulation. The latter observation might be explained by the the fact that the algorithm minimalizes the cross-entropy, which is the negative loglikelihood: there is no unique ensemble of parameter values for a certain value of the loglikelihood. The following solutions are equivalent (equation 8), for any value of ϕ and ψ.

$$\sigma(a \cdot (\theta - b)) = \sigma(\frac{a}{\phi} \cdot (\phi(\theta + \psi) - \phi(b + \psi))) \tag{8}$$

No that this means that the model is not identified - in the sense that there is no unique maximum for the loglikelihood. The absolute parameter values thus do not have any meaning. Neither do fractions of the parameter estimates for b and θ mean anything as translations can be made by adding constants to both b and θ. The interpretation of the model is thus limited to the order of the parameter estimates. This is not a problem as the teachers who will have to interprete the model are only interested in the order of the parameters and in the probability estimates. We should thus check whether the order of the parameters is preserved in recovery simulations. The quantity which expresses this best is correlation.

We observe that the highest and the lowest estimates for b in the simulation in Figure 15 are single points far from the rest. These points could be outliers. When we look at Figure 12, we observe that for low values of θ, θ seems to be overestimated, and for high values underestimated, for a simulation of 25 items. For a simultaneous item and student parameter

[16] Note that this heuristic sets the values of b between -3 and 3. The limits of our distributions to generate the data with follow the same recommendation. This yields similar results to the distributions. When we put a different heuristic on b, the results are similar but for a linear transformation as intended in equation 8.

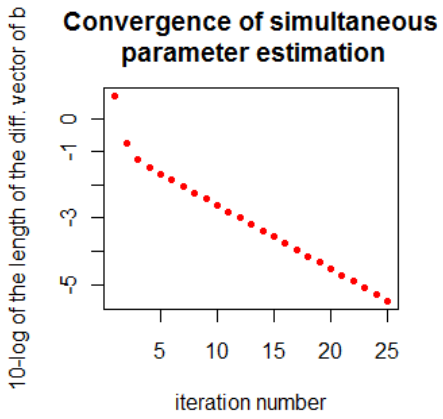


Figure 14: The convergence of a simultaneous parameter recovery simulation with 250 items and 250 students, a and b uniformly distributed between 0.8 and 1.8 and between -3 and 3 respectively, θ uniformly distributed between -3 and 3. The y-axis displays the 10-log of distance between the vector of b estimates at iteration t and $t-1$. Note that this vector has length 250.

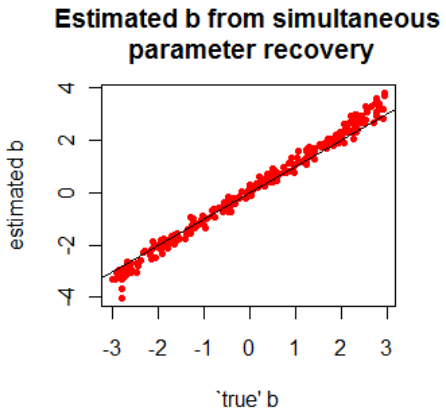


Figure 15: The estimated value of b as a function of the true value of b. Estimated in a simulation with 250 items and 250 students a and b uniformly distributed between 0.8 and 1.8 and between -3 and 3 respectively, θ uniformly distributed between -3 and 3.

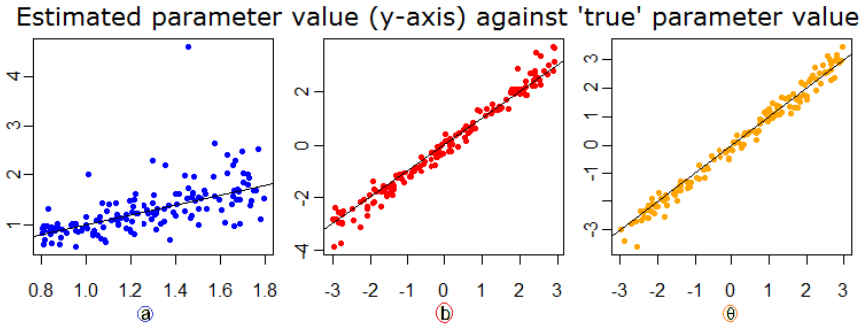


Figure 16: The parameters a, b and θ recovered from a 150 student, 150 items simulation. The line represents a line through the origin, with slope one.

recovery of 150 items and 150 student, this does not seem to be the case, as can be seen in Figure 16. The opposite seems more likely. What we also observe is that the estimates for a are more unreliable than those for b and θ as the correlation between a and estimated a was 0.62, while the correlations between b and estimated b and θ and estimated θ were both 0.99 when rounded to two decimal places.

4.12 Learning item and student parameters simultaneously in small datasets

For smaller datasets the algorithm has trouble converging due to two problems. For a low number of outcomes per student or per item, there is a high probability of there being an item/student with outcomes of only 0 or only 1, or, that there is an item which has outcomes that are linearly seperable using θ. In the first case we would get infinite difficulty or ability. In the second case we would get infinite discriminativity.

To bypass the problems with convergence in smaller datasets, we use discrete loglikelihood optimization instead of IRLS when the amount of data is small. We created this algorithm ourselves. The algorithm we obtain initializes a and b for all items (with the same educated guess for b as described before and 1 for all a), and then repetitively finds the θ for all students that has the highest value for the loglikelihood between two limits with a certain stepsize[17], and then the same for all combinations of a and b for all excercises. Note that when obtaining the optimal value for θ only

[17]The stepsize is a parameter of the algorithm.

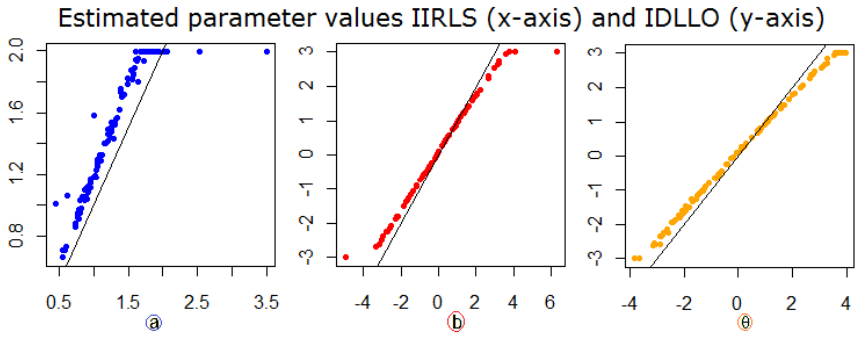


Figure 17: The parameters a, b and θ recovered from a 100 student, 100 items simulation for the same data by IDLLO (on the y-axis) and IIRLS (on the x-axis). The line represents a line through the origin, with slope one.

one student is considered at a time, as all student parameters for different students are independent given the item parameters, and vice versa. This algorithm is considerably slower the one using IRLS, as lots of processor time is lost examining parameter values that are far from the optimum value. Furthermore, no preciser estimates can be made than the stepsize.

We implemented this algorithm in JAVA, and ran a simulation in order to compare it to the IRLS. For this purpose we ran a simulation with 100 students and 100 items, using a uniformly distributed between 0.8 and 1.8 and b and θ uniformly distributed between -3 and 3. The same data was used for the iterated IRLS (IIRLS) algortihm and the iterative discrete logilikelihood optimization (IDLLO) algorithm. IDLLO used steps of 0.01 for all parameters. a was tried between 0.5 and 2.0 and θ and b were tried between -3 and 3. IIRLS took 1.58 minutes to converge, and IDLLO took roughly 78 minutes, on the same machine [18] for 20 iterations, after which it was stopped. When we compare the estimates for a, b and θ (Figure 17), we observe a systematic difference in all parameters, and the effect of capping the values of the parameters at a maximum (at the top of the plots). Note that the lower cap is almost never reached. At first sight, the two algorithms output different parameter values. The probability estimates however, are similar. This is because the model is not identified (see equation 8). From Figures 18 and 19 we observe that most outcomes for IDLLO and IIRLS are identical with respect to the probability estimates. The estimates that

[18] An ACER Aspire M3300, with an AMD Phenom II X6 1035T processor of 2.60, and 4GB RAM, running Windows 7 64-bit. Only one core was used.

are further off, seem to be caused by the maximum and minimal values for the parameters that IDLLO sets. Both algorithms make different estimates for the probability of the outcomes, but for the 100 times 100 probability estimates for the outcomes, only 4.8% of the estimates would yield a different prediction (0 or 1) for the actual (dichotomously scored) outcomes. Note that this does not mean that the accuracy of the predictions are 95%, because the stochastic component is not yet taken into account, just that the predictions are 95.2% the same as when we would use the original parameters as a model. Furthermore we observe that the region where the estimated and the original parameters would disagree is closer to the middle, where the predictions are less certain to begin with (as the probabilities are close to 0.5). What the effect of this is in terms of accuracy will be explored in the section 4.14.

The last thing we check is the performance of the algorithm for small amounts of data. In chapter 6 we will show that the data we gathered contains 23 rules, which in the IRT context can be seen as items. We were only able to gather data from 15 students. In order to check the performance of IRT parameter estimation on the amount of data we have, we did a simultaneous parameter recovery for 23 items and 15 students.[19] For this small amounts of data it proved infeasable to do IIRLS due to the problems previously discussed (only positive or negative outcomes and linear seperability). IDLLO does not take too much time, due to the small amount of data so only IDLLO was run. In the first recovery simulations we find that the probability estimates when using this method were rather extreme (close to either 0 or 1) compared to the probabilities that were calculated with the original parameters - the parameter we generate the outcomes with. The parameter estimates for a also tend to be more extreme (8 out of 23 estimates for a were at maximum permitted discriminativity 3.5 which is an unlikely parameter value). This seems to indicate overfitting.

4.13 Intermezzo: the Rasch model

In the work on parameter recovery Baker and Kim [BK04] take note of a suggestion in literature that the Rasch model (1PL model), may yield better difficulty parameter estimates, even when the original data was generated by the 2PL model. To test this hypothesis we estimated the parameters three times. In the first estimatation the range of a was restricted to 0.5 to 3.5, and could take any value with two decimal places between these two

[19] Note this is not the data we gathered, but data we generated from the 2PL model with original parameters, distributed uniformly, equal to the previous simulations.

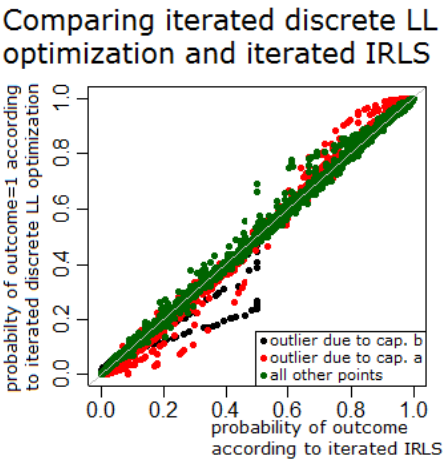


Figure 18: The probability estimates for the 10000 outcomes (from a 100 students, 100 items simulation) of IDLLO plotted against those of IIRLS. The black points are those estimates made by a capped value of a (2.0), the red points are those estimates made with a capped value of b, and the green points are estimates for non-capped parameters. Green is plotted over red, which is plotted over black, so note that there are many other colored points under the green points. The gray line is a line going through the origin, with slope 1.

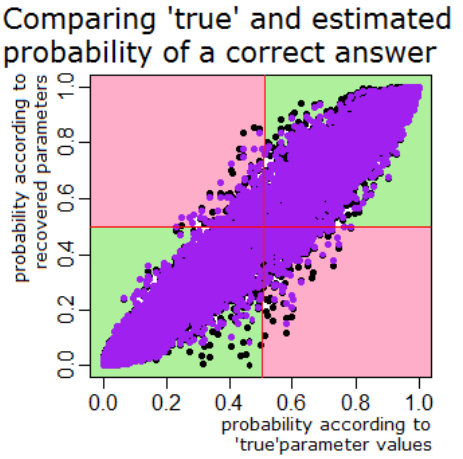


Figure 19: The probability estimates for the 10000 outcomes (from a 100 students, 100 items simulation) of IDLLO (purple) and IIRLS(black) plotted against the 'true' values of the parameters. IDLLO was plotted over IIRLS. The green quadrants are the regions where the 'true' parameters and the estimated parameters would agree in terms of the (dichotomous) prediction of the outcome, the red quadrants is where they would disagree. Although not clearly visible in this plot, the point density around the middle is much lower than the density near the extremes. The number of points where the original parameters and the estimated parameters disagree was only 478 (IDLLO) and 480 (IIRLS).

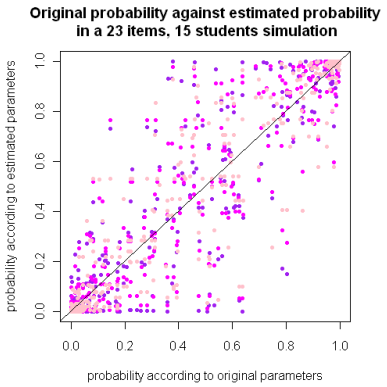


Figure 20: The probability estimates using IDLLO plotted against the probabilities according to the original parameters. The purple dots are the probabilities using a's lying between 0.5 and 3.5 tested in steps of 0.01, the magenta dots are the probabilities using a is one of 0.5, 1.0 and 1.5, and the pink dots are the probabilities using a is 1.0.

limits (so the values between 0.5 and 3.5 were tested in steps of 0.01. In the second estimation a could either be 0.5, 1.0 or 1.5. In the last estimation a could not vary and its value was 1.0 (this is the Rasch model). All the estimates were made on the same data, using the IDLLO algorithm. The data was generated using true a uniformly distributed between 0.8 and 1.8 and b and θ uniformly distributed between -3 and 3. The original parameters were stored as well. When we plot the probability according to the original parameters, against the probabilities estimated using the three possible parameter recoveries (Figure 20), no significant difference in quality of the estimates is apparent. Also the count of the outcomes where the original parameters and the estimated parameter values disagree, is not significantly different; 43 (12.5%) for the first estimation, 45 (13.0%) for the second and 44 (12.8%) for the Rasch model estimates.[20]

[20]Note that a convienent metric like the coefficient of determination (R-squared), for the quality of a model with respect to the training data set does not exist. Therefore we chose counting the difference in classification according to the original parameter values and the estimated parameter values as a quality measure for the estimated model with respect to the training set.

4.14 2PL as a predictor

So far we have described the estimates of parameters and probabilities with respect to the distribution generating the data. From this we get an idea of uncertainties in the parameter estimates and sensitivity to the amount of data. However this does not yet say a lot about the usefulness in the context of e-tutoring systems - only which estimates are (un)reliable. In e-tutoring systems we can use IRT for two purposes; using θ as a measure of competence to report to the student, in self-tests or in learning modes, and in order to select a next appropriate excercise according to a teaching strategy. In self-tests where no feedback is provided the parameter estimates do not differ from standard parameter recovery. When we do provide feedback however, the competence estimates should change over time because the student learns while/in between of producing answers. In this section the opportunities for using classic IRT for e-tutoring systems will be explored by simulations. We will do simulations to check the accuracy of prediction. We can use the accuracy of prediction to base a teaching strategy on: e.g. we do not want to give a student too many exercises which he/she cannot solve and not only exercises which a student has a near to one probability of solving in order to keep it challenging.

We define the process of predicting outcomes and calculating accuracy. This process consists of three steps:

1. Estimate θ with all known outcomes and item parameters by means of IRLS.

2. Predict the next outcome using the θ estimate and the item parameters of the next item.

3. Compare the predicted result with the actual result, add the result to the known results and repeat from step 1, until there is no more data.

First we will evaluate the estimation of θ, from a sequential and sensitivity perspective, and secondly we will look at the accuracy obtained this way, again from a sequential point of view.

4.14.1 Estimating θ on the fly

If we assume there is no learning effect and the item parameter estimates are sufficiently accurate, we ought to get more and more information as a student does more and more exercises and thus a better estimate of the

student competence. This assumption is acceptable for tests, because we assume that during test any learning effects are negligible.

First we look at the estimates for θ using the IIRLS estimates of the item parameters from the previously performed 100 items by 100 students simulation (Figures 17-19). The items are presented in the same sequence several times, to a simulated student with constant competence. The outcomes were generated independently each sequence, leading to different estimates of θ. Six typical results are given in Figure 21. We observe that in the region where θ changes least, between the red lines in the figure, the difficulty of the items were all around -2, and the outcomes all positive. In the areas where θ does change a lot the item parameters were closer to the true value of θ. Furthermore we observe that later in the sequence, the estimate of θ seems more stable. This is to be expected, as the relative amount of new information versus already known information goes down.

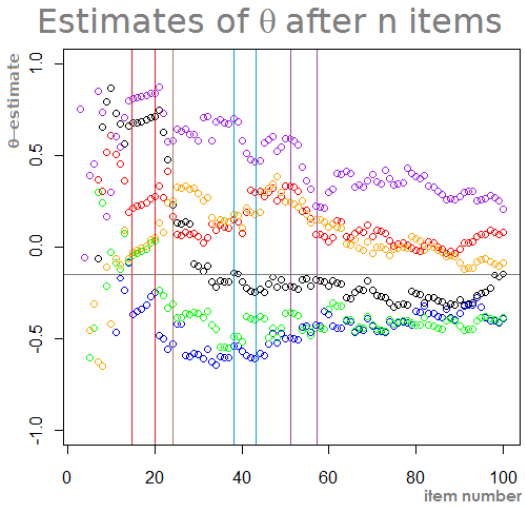


Figure 21: θ estimated with six independently generated sequences of outcomes. The horizontal (grey) line represents the true value of θ. The red lines denote the sequence between 15 and 20 and the next orange line is at 24. The first is a sequence of interest as θ seems to be rather stable. The second is a region which contrasts this with larger differences. The blue and purple lines also contain parts of the sequence where θ changes a lot.

When we look at the sensitivity to an item we expect that the distance between the current estimate of θ and the difficulty of the next item will be of influence. When we look at a logarithmic plot (Figure 22), this seems

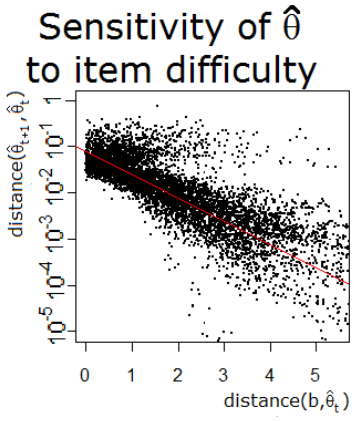


Figure 22: The change in a θ estimate as a result of an item with a difficulty with a certain distance to the current θ estimate. The parameter estimates from the 100 by 100 simulation have been used, but the outcomes have been generated anew. The red line has slope -0.5058, and an intercept with the y-axis of -1.0886, which were the result of fitting a line using the `lm` function in R.

to be the case indeed. Also we see some points like an "arm" sticking out above all the other points. These are probably the instances were the prediction would have been wrong. (When we removed these points from the data, most of the arm, and the highest points in general disappeared. The average effect of an item can be fitted into the following formula $\|\hat{\theta}_{t+1} - \hat{\theta}_t\| \approx 10^{-1.0886-0.5058\|b_t-\hat{\theta}_t\|}$, which is an exponential decay function. The explained variance (R-squared) of this model is however only 15%. When we include whether or not the prediction was correct, the explained variance was 20%. Note that, unlike "wrong prediction", we can influence the $\|b_t-\hat{\theta}_t\|$ by selecting a different exercise for a student when he/she is working in an e-tutoring environment.

What we cannot know (while measuring a new student), but ought to have an effect is the "wrongness" of the current estimate. When the estimate is further away from the value it should be, the next items are likely to pull the estimate towards the true value. If the estimate is equal to the true value however, the new outcomes are most likely to be in line with the expectations, which means that the value of the estimate should not change (a lot).

4.14.2 The accuracy of predicting the outcomes using IRT

From the competence estimate we can predict an outcome for the next exercise. When $\theta_{curr.est.} > b_{next}$ we predict an a positive outcome ($o_{next} = 1$) and otherwise we predict a wrong answer $o_{next} = 0$). We have seen (e.g. Figure 13) that estimates for the competence of a student θ tend to be less accurate when the competence is on either end of the distribution of competences of the group of students and the item difficulties (which are the same in our experiments). The predictions of outcomes are however most uncertain when $\theta = b$ (in which case the probability is 0.5), which would imply that for predictions on the edges of the competence/difficulty spectrum the estimates are more certain. Also we have seen that the probability estimates after item recovery agree almost completely with the original probabilities in terms of classification (95.2% the same predications for 100 students and items, and around 83% for 23 items and 15 students). The latter of course does not mean that those percentages are the attainable accuracies.

When we use the parameter estimates for the 100 by 100 simulation, and estimate θ as we go, we see that the highest accuracies are around the more extreme values of θ, for both the "real" value of θ of a student, and the estimates that are made. Note also that the estimates of θ are spread out more than their real values. For more extreme values of b the accuracy seems to be higher as well. Figure 23 shows several views on accuracy.

Looking at the accuracy after the i-th item, we observe that for this case the accuracy until about the 20th item seems to increase, and then is spread around 84%. When we fit the accuracy as a line from item 3 to 20, and use the line to estimate where the average accuracy is attained, this seems to be the case around the 18th item. The mean accuracy over all items is 83%.

Let us compare this to the theoretical accuracy that would be expected over the entire interval. We can calculate analytically (equation 9) for θ uniformly distributed between -3 and 3, what the expected accuracy is when $a = 1$ and $b = 0$. We calculate this as the probability of a correct prediction, integrated over the distribution of θ. This distribution is a uniform distribution between -3 and 3.

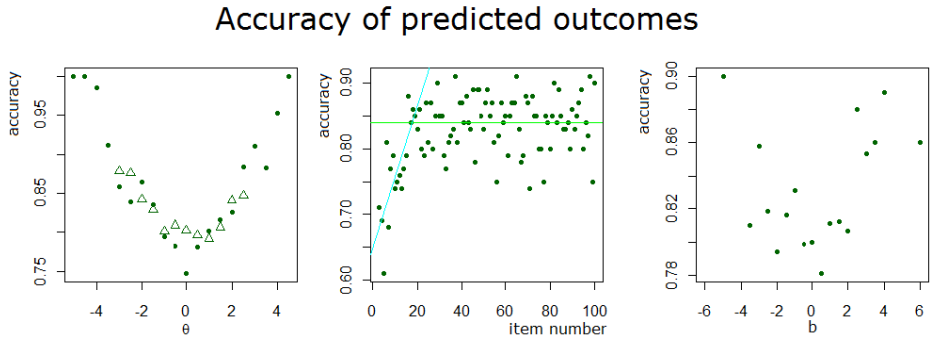


Figure 23: The accuracy as a function of several variables. In the left plot: the estimated (dots) and "real" (triangles) θ from the 100 by 100 simultaneous item and student parameter recovery previously performed. In the middle plot:The number of items on which the predictions are based (in the same order as the parameter recovery). In the right plot: the estimated b from the parameter recovery. The trends are easily observable, the accuracy is lowest for θ's around 0 (note that the item difficulties are spread between -3 and 3), the same seems to hold for b (in this case the original student parameters are spread between -3 and 3). The accuracy seems to increase with the number of items the predictions are based on until around the 18th item, and stabilizes afterwards. Note that for some items the accuracy is lower. This is caused by a combination of the item difficulty and the item discriminativity.

$$
\begin{aligned}
& 1/6 \int_{-3}^{3} max \left\{ \frac{1}{1+e^{-1(\theta-0)}}, 1 - \frac{1}{1+e^{-1(\theta-0)}} \right\} d\theta \\
&= 2/6 \int_{0}^{3} \frac{1}{1+e^{-\theta}} d\theta \\
&= 1/3 \int_{0}^{3} \frac{1}{1+e^{-\theta}} d\theta \\
&= \ln(1+e^3) - \ln 2 \\
&\approx 0.785
\end{aligned}
\tag{9}
$$

Now let us find the worst case for the simulations we have performed. In the simulation we have used a uniform distribution between 0.8 and 1.5 for a. The worst case would thus be $1/3 \int_0^3 \frac{1}{1+e^{-0.8\theta}} d\theta \approx 0.75$, approximated numerically. The best combination of a and b would be $a = 1.5$ and $b = \pm 3$. The integral would then be $1/6 \int_{-3}^{3} \frac{1}{1+e^{-1.5(\theta+3)}} d\theta \approx 0.92$. We could further narrow down the results by integrating over the interval of a, which would result in the integrals (10) for the best case, and (11) for the worst case.

$$
\frac{1}{6 * 0.7} \int_{-3}^{3} \int_{0.8}^{1.5} \frac{1}{1+e^{-a(\theta+3)}} da \, d\theta \approx 0.90 \tag{10}
$$

$$
\frac{1}{3 * 0.7} \int_{0}^{3} \int_{0.8}^{1.5} \frac{1}{1+e^{-a\theta}} da \, d\theta \approx 0.80 \tag{11}
$$

From the simulation results, we calculated the accuracy for different intervals of b with an interval width of 0.5. The best value was found for an interval of b was indeed 0.90 for $[-5.5, -5)$, which has a hundred data points. The worst value for a b interval we found in the interval $[0, 0.5)$, which has an accuracy of 0.78 and 900 data points.

We also calculate the accuracy for different intervals of θ with a bin-width of 0.5. The best value was found for an interval of θ was 0.99 for $[-4.5, -4)$ with 136 data points.[21] The worst value for θ was found in the interval $[-0.5, 0)$, which had an accuracy of 0.75 and 1110 data points.

For the smaller simulation (15 students, 23 items) we also ran the prediction while updating the θ estimates after each item, as can be seen in Figure 20. The overal accuracy, after the third item, is 0.76. When we look at the accuracies graphically (Figure 24) we see that the spread of the points seems to be less orderly. Some points are generated by five points or less, which has to be ignored due to uncertainty (in the figure these are striked

[21] There are in fact bins with an accuracy of 1, but with less than ten data points, so we ignore those.

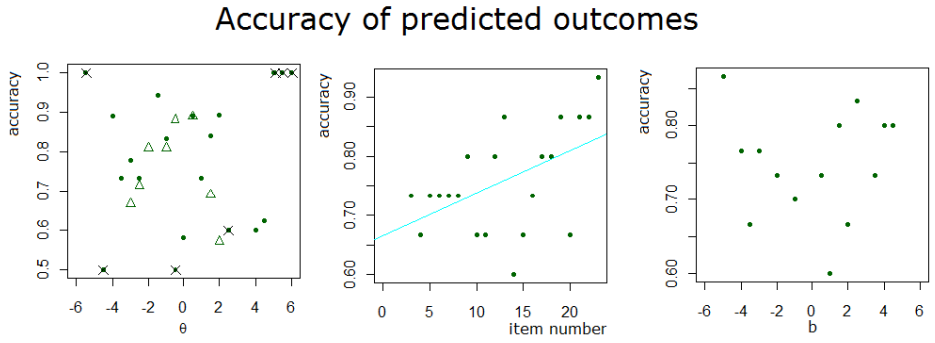


Figure 24: The accuracy of predicting the outcomes using IRT in a small simulation as a function of several variables. The estimated (dots) and "real" (triangles) θ come from the 23 items by 15 students simultaneous item and student parameter recovery. The number of items on which the predictions are based (in the same order as the parameter recovery), and the estimated b from the parameter recovery are presented in the middle and right plots. The trends are less clear than for the 100 by 100 simultaneous item and student parameter recovery. It is unclear where the accuracy is lowest for θ and b. For "real" θ the accuracy seems highest around 0, which is strange because the accuracy is worst for θ around 0 for the larger simulation. This could still be a random effect, but another explanation that seems plausible is that the certainty of the student parameter is highest around 0. The accuracy seems to increase with the number of items the predictions are based on. We cannot determine whether the increase in accuracy end at the end of this graph, but as the accuracy is around 0.8 this is likely.

out). Something that seems to be very different from the case of the larger simulation (Figure 23) is the shape of the accuracy pattern for "true" θ, as is given by the triangles in the utmost left frame. For the larger simulation, more extreme θ tend to be better, while for the small simulations the more extreme values of "true" θ seem to lead to lower accuracies. One possible explaination for this, is that the more extreme values of "true" θ tend to get over- or underestimated too far, which leads to bad results. This suspicion seems to be confirmed when we check the θ estimates.

The worst values for bins with more than 5 points are: $-0.5 < \theta < 0$ with an accuracy of 0.58 with 79 data points and $0.5 < b < 1$ with an accuracy of 0.60 with 15 data points. The best bins are $1.5 < \theta < 2$ with an accuracy of 0.89 with 46 data points and $-5.5 < b < -5$ with an accuracy

of 0.87 and 15 data points. For the item sequence the accuracy seems to be rising, but we cannot tell whether the accuracy has reached its maximum or not.

4.15 Towards a student model for rule based e-tutoring systems - lessons learned from the 2PL model

In this chapter sofar we have introduced item response theory (IRT). IRT is a theory which relates competence to the probability of a correct answer for an item (an exercise), and thus seems a suitable basis for use in e-tutoring systems. We have selected the two parameter logistic ogive variant of IRT (2PL), which is often used in practice. 2PL contains the parameters: difficulty, discriminativity and competence.

From parameter recovery simulations using the 2PL model we have seen that the estimates of the parameters competence θ, and difficulty b, are less reliable at the edges of the distributions. We have performed parameter recovery simulations using b and θ distributed uniformly between -3 and 3. When the parameter to estimate (b or θ) goes over the edge of the distribution of the other, the standard deviation in the estimates seems to "explode", and becomes very unreliable. (Even with 100 simulated students the standard deviation around ± 4 was around 0.6, which is very high, and extrema of around 1.4 as a standard deviation in a sample of 15 have been observed. For "normal" values of both b and θ the standard deviation decreases exponentially with the number of students (for estimating b) and items (for estimating θ). The standard deviation in the discriminativity parameter of an item, a, follows that of the standard deviation in difficulty b, and thus mostly depends on the value of b.

For simultaneous item and parameter recovery, we observe that the parameters can be recovered well simultaneously. However, there is no guarantee that the parameters will approximate the original parameters, as there are many ensembles of parameters that are equivalent. (We have observed in simulations that the estimated parameters in b and θ can be spread out more than the original b and θ, while the recovered a values seem to be spread out less than the original values of a.) Furthermore we observe that the estimates in a seem to be less reliable than those of θ and b. As prediction of the outcomes depends only on θ and b, this is still okay for prediction purposes.

We have seen that for small datasets/simulations our standard heuristic algorithm of iterative iteratively reweighted least squares (IIRLS) does not work well, and we have thus defined a discrete algorithm which approx-

imates IIRLS: iterative discrete loglikelihood optimization (IDLLO), and have proven it leads to comparable results (largely the same in terms of probability estimates) for larger datasets (though IDLLO takes a lot longer to run).

We have simultaneously recovered the item and student parameters in simulations, and shown that this works rather well, even for small amounts of students (15) and items (23). In roughly 1/8-th of the probability estimates, the original and estimated parameters would "disagree" in terms of the prediction of the outcome. This happens most around probability estimates of 0.5 where the certainty is lower anyway.

We have tested the notion that the Rasch model (1PL) would be better than 2PL at estimating the probability distribution even when the data was generated by a 2PL distribution - as mentioned in literature. We have, for small datasets, however found no evidence for this in terms of probability.

When we look at estimating θ while the student is performing exercises (in simulations that is), we have seen that the estimate of θ is on average most sensitive to items for which the difficulty is close to the current θ estimate. The θ value changes a lot when the prediction for the outcome would have been wrong - it is however impossible to know this before we have offered an item to a student, and is therefore not useful information for teaching strategies.[22]

The accuracy of prediction after simultaneous student and parameter recovery was determined experimentally. We also looked at the theoretical expectation for accuracy. We have seen that for a 100 items and 100 students simulation, the accuracies were close to the theoretical expectations for accuracy around several parameter values. The overall (experimental) accuracy over all outcomes of this simulation was 83%. For a small simulation of 23 items and 15 students, the overall accuracy dropped to 76%. The accuracy of this simulation was lower over the entire distribution of the possible parameter values. We have observed that the accuracy went up as the number of items on which the estimates are based went up. In the larger simulation, we have seen that accuracy stops increasing after a while, for

[22]If we were to define a teaching strategy, choosing exercises with a difficulty b close to the value of θ would seem a natural choice from a prediction point of view (it influences the competence estimate most). Furthermore this would also be the most natural choice when we look at Vygotsky's theories on ZPD [Vyg78], which states that you should offer a student an exercise which is just above the present skills of the learner. So one should offer the learner exercises which he can solve with a little help. A probability of success of (around) 0.5 which would be the probability of succes when $\theta = b$, would fit this description, als the student has a 0.5 probability of not completing the item on his/her own. One could choose items with difficulties close to the θ value, or even a bit higher.

the small simulation there probably were not enough items to observe this phenomenon.

We conclude that 2PL, though less accurate, can be used as a model, even when the number of students, and items is small. It can be used as a predictive model for the outcomes. 2PL can also be used as a descriptive model for the relative difficulty and discriminativity of items and the relative competence of students. 2PL assumes that while a student does exercises his/her competence does not increase. For e-tutoring systems this is an important limitation. The use of standard IRT is limited to (self-)tests in this context. When we supply feedback while the students solve the exercises, we have to incorporate learning into the model.

We will take the following points from the 2PL model with us, while creating our student model for rule-based e-tutoring systems:

- When recovering parameters we will have to take into account that the parameter estimates can become very unreliable when the student competence θ is outside of the range of the difficulties of the items, and vice versa.
- The estimates of discriminativity are less reliable than those of competence and difficulty. We will have to check this for any extensions to the 2PL model.
- To estimate parameters in small data sets we created the IDLLO algorithm, because IIRLS does not work.
- An accuracy of around 76% was achieved for a small prediction simulation (23 items, 15 students). This figure we will compare to the performance of the student model for rule based e-tutoring systems.
- We can reuse or extend most of the algorithms we implemented for parameter estimations, simulation and prediction, as we base our student model on 2PL.
- The model, 2PL, is not identified. Extensions to this model will not be either. We can thus only interpret the order of the parameter estimates.

4.16 Extending the 2PL model to a student model for learning in rule based e-tutoring systems

In this section we will describe the possibilities for extending IRT for use in rule based e-tutoring environments. In rule based e-tutoring environments

a student is presented with a problem, which can be solved stepwise. Each step leads to a new situation in which the student should determine:

1. Which rule(s) can and should be applied (identifying possibilities)
2. What the rule he/she wants to apply is (remembering the rule)
3. What the next situation looks like (applying the rule)

The environment can only determine which rule the student applied; often automatically, sometimes by using the judgement of an expert/teacher.[23] We can thus not separate these three points by looking at the data. We assume that each situation in which a rule can be applied is equally difficult and discriminative, so that we can see a rule as an item in the sense of item response theory.[24] This assumption allows us to model a student as a set of "rule competences".

We attempt to model student learning. The most straightforward model is the model in which the competence of a student increases linearly with the number of times a student encounters an item. This model is presented in equation 12.

$$P(o_{r,s,t} = 1) = \frac{1}{1 + e^{-a_r(\theta_{r,s,0} + \eta_{r,s} t_r - b_r)}} \tag{12}$$

In this equation r denotes the rule, s the student, and t the t-th time the student encounters the rule. $\theta_{r,s,0}$ is the start competence (at $t = 0$) of a student for a rule. $\eta_{r,s}$ is the increase for a student for a rule, per encounter. b_r is the difficulty of a rule, and a_r the discriminativity. Note that this model would mean that for each rule a student can have a different start competence θ_0, and a different learning speed η.

We explore whether it is possible to use such a model using simulation, that is to say, can we recover student parameters within acceptable uncertainty? We do a preliminary parameter recovery simulation to test whether this is possible. We use item parameters $a_r = 1.0$ and $b_r = 0.0$ and recover the student parameters. These item parameters are an arbitrary value pair. The student parameters are set to $\theta_{r,s,0} = -2.0$ and $\eta_{r,s}$ we vary. We use these same parameters 15 times, and simulate the student doing the rule n

[23] In `ideas` many so-called buggy rules are known [GHJ08], but of course there will always be mistakes that the design of the program has not foreseen. The goal in the design is to be as complete as possible, of course.

[24] This is of course a simplification, but for the logic domain, it was our expert opion as teachers in computer science.

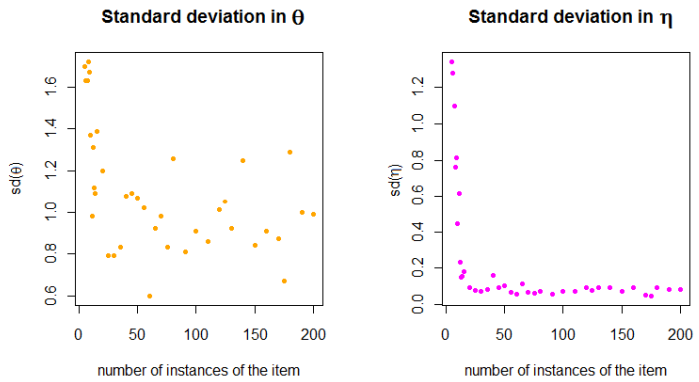


Figure 25: When the start competence θ_0 and the increase in competence per use of a rule, are independently estimated per student and rule, only one series - for a certain exercise, can be used for estimation. For one item with difficulty $b = 0$ and discriminativity $a = 1$, the points in this graph show the standard deviation in the recovery of the student parameters $\theta_0 = -2$ (start competence) and $\eta = 0.2$ (increase in competence per item view).

times. We generate outcomes based on the probabilities according to the parameters and equation 12 stochastically. For $\eta_{r,s} = 0.2$ and $n = 10$ the mean of the estimates were close to the true value of the student parameters, but the standard deviations were very large: about 1.4 for the start competence, and 0.44 for $\eta_{r,s}$. For these parameters and this small amount of data, the model is thus not viable to use in practice. When we raise the start competence to -1.0 the standard deviation of the start competence does not change. The standard deviation in $\eta_{r,s}$ becomes slightly lower, around 0.3, which is still not viable. The reason the uncertainty decreases is probably because now the transitional part around a probability of a correct answer of 0.5 is better represented in the sequence. When we set the start competence back to $\theta_{r,s,0} = -2$ and increase the number of outcomes for the rule we obtain the graph in Figure 25. We observe that the standard deviations in both student parameters initially decrease, and then stabilize. The values of the standard deviation stabilize at rather high levels; around 1 for the start competence, and around 0.08 for the increase in competence per item view. All this time the sample mean of both estimates of student parameters are around the true values.

We conclude that estimating start competence and increase in competence per rule is not feasable - due to high standard deviations - even when we have a lot of data per student. Other factors that support this conclusion

are that this method is very sensitive to wrong estimates of b, and probably also of a. The only option we have is to make the model more restricted. For this we have several options:

1. The start competence per rule can be different, but the learning speed is always the same for one student.

2. The start competence is the same for all rules for a student, but the learning speed can vary.

3. The start competence and the learning speed do not depend on the rule, but are properties of a student only. (Within one domain.)

Experts in our domain immediately discard option 2, so we will not further consider it. The difference between options 1 and 3 is that there is either a domain, or a rule specific start competence. The domain specific model is more restricted (fewer parameters) which has the disadvantage that it might be an oversimplification. The advantage of a domain specific start competence is that we could predict the development of a student learning a new rule in the domain, given the difficulty (and discriminativity) of the rule, before we offer the rule to the student. In the rule specific model we should first learn the start competence of a student for a rule, after which we would be able to predict the number of steps a student would need to reach a certain competence level. Both models, would provide new and useful information in the context of e-tutoring systems, as the system can automatically select a sufficient number of exercises for a student to reach a certain competence.

Note that both models are probably a simplification of reality: they do not model the case where some students would learn all rules slowly except for one rule for which the student would suddenly see the light, while other students learn the same rule slowly as well.[25] Also this model does not account for previous knowledge for a specific rule. In the logic domain this could occur by a portion of the students having been told the "do not eat and drink" example. This sentence could be interpreted logically as "do (not (eat and drink))" which would mean "do ((not eat) or (not drink))" and can be discarded as a friendly warning meant to avoid students from choking. This example corresponds to one of DeMorgans laws in the logic domain.

[25] If all, or at least most, students would at some point "see the light" for a rule, it would just mean that the model would be correct when the rule has a very high discriminativity ("you either get it, or you don't").

What this model also does not contain is "transfer knowledge" - what a student can learn about one rule, from performing another rule. Some rules might be highly correlated (e.g. the two variants of DeMorgans law), or the global understanding of a student of the domain increases. The latter could be modeled by using an extra parameter, similar to η, modelling the increase in competence of the student by each item presented in the domain.

4.16.1 Student model 1: learning speed per student, start competence per student per rule

First, let us look at the possibilities of the first option: the start competence per rule can be different, but the learning speed is always the same for one student. This models previous knowledge a student might have about a rule. The formula for the probability function would be as follows:

$$P(o_{r,s,t} = 1) = \frac{1}{1 + e^{-a_r(\theta_{r,s,0} + \eta_s t_r - b_r)}} \tag{13}$$

Using discrete maximum likelihood optimisation to recover the parameters $\theta_{r,s,0}$ (distributed uniformly between -2 and 2) and η_s (distributed uniformly between 0.02 and 0.50) in 100 students, 100 rules and 10 instances per rule. The rule parameters are distributed as in the previous simulations: a has a uniform distribution between 0.8 and 1.5, b is uniformly distributed between -3 and 3. We observe that both $\theta_{r,s,0}$ and η_s are recovered with a high dependency on the original parameters. However, the variation in $\theta_{r,s,0}$ is rather high.

Note that when the rule parameters are known, the model is identified. When we subtract the original from the estimate (here we use the original parameter as a "model" for the estimate), and calculate the mean and the standard deviation (around the "model"), the mean is close to zero, but the standard deviation was 1.89. This number is mainly caused by items with a maximal, and minimal estimate for difficulty ± 4. This happens when the outcomes of such an item are all positive, or all negative, which happens often when the number of instances per item is only 10. When we remove these items, the standard deviation in $\theta_{r,s,0}$ remains 0.79. The original values of this parameter were uniformly distributed between -2 and 2 in this simulation. A standard deviation this high is too much for practical use. For the data we gathered, (25 items, at most 14 instances per rule, and 15 students) this model is thus not useful.

When we increase the number of instances per item from 10 to 50, the standard deviation in $\theta_{r,s,0}$ for non-capped data points remains 0.69. Even

though the correlation between the original and the recovered parameter values is 69%, the start competence cannot be estimated with enough certainty to give a reasonable estimate of the number of exercises the student would need.[26] This disqualifies this model as a predictive model.

4.16.2 Student model 3: learning speed and start competence per student (extended IRT)

Secondly, let us look at the possibilities of the third option: the start competence and the learning speed do not depend on the rule, but are properties of a student only. (Within one domain.) This does not model previous knowledge a student might have about a single rule, but does model previous knowledge about the domain.[27] The formula for the probability function is:

$$P(o_{r,s,t} = 1) = \frac{1}{1 + e^{-a_r(\theta_{s,0} + \eta_s t_r - b_r)}} \tag{14}$$

The recovery of the student parameters generated from this model gives much better results. For 100 students, 100 rules and 50 instances per rule, the results are shown in Figure 26. The results of the previous simulation (Figure 25) for θ are also displayed as the gray points on the background. The correlation between the original parameter value and the estimated value was 0.99, It is important to note that although the student parameter recovery is more successful for this latter, more restricted model, it does not follow that this is a better model to describe reality with. What these simulations do say is that the last model is a more reliable model: the original parameter values of the start competence describe better what the estimated values of the start competence will be (a correlation of 0.99 versus 0.69).

We also performed a small simulation, with 23 rules, 15 students, and 8 instances per rule. The spreads of the parameters were as in the larger simulation. The results are visually displayed in Figure 27. We observe that there is still a small spread around the line through the origin with slope one. When we use the original parameter to explain the variance in the

[26] If η_s is 0.2 the number of steps that is needed to reach a compentence level differs by about 3.5 at an error in the estimate of $\theta_{r,s,0}$ 0.69.

[27] This would seem an reasonable model in situations where certain rules are naturally acquired after one another. E.g. in basic calculus the student would acquire addition, substraction, muliplication and division, in that order. In this model addition would be the easiest rule (in terms of b) and division would be the hardest. As the overal competence is on the same scale as item/rule difficulty the rules a student already knows about will be visible from their start competence.

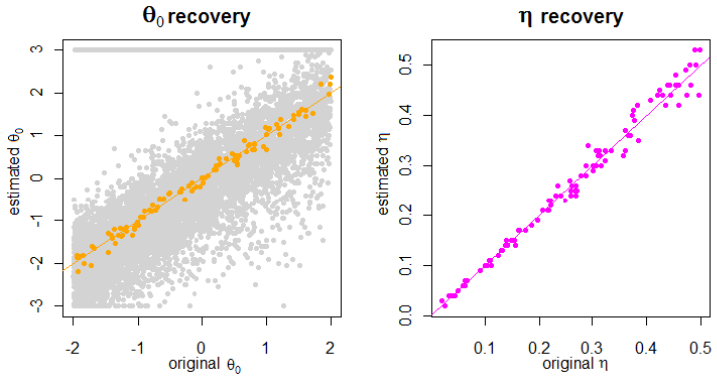


Figure 26: The recovery of the student parameters in 100 rules, 100 students and 50 instances per rule. b and a were both uniformly distributed. b between -3 and 3 and a between 0.8 and 1.5. $\theta_{s,0}$ had a uniform distribution between -2 and 2, and η_s between 0.02 and 0.50. The grey dots in the image for $\theta_{s,0}$ are the estimates that were achieved in a similar simulation (with the same parameters), but then using the model where $\theta_{r,s,0}$ depends on the rule as well. The contrast between this less restricted model and the model using $\theta_{s,0}$ independent of the rule, is clear. Note that the lines are no fits, but lines through the origin with slope 1.

estimated values the explained variance (R-squared) was 91%. The square root of the variantion of the estimated parameter values minus the original parameter values (standard deviation around the original parameter values) was 0.31. The correlation between the original parameter values and the estimated parameter values was 0.95. Note that this is still much better than the larger simulation using a start competence per rule per student. For η the standard deviation after subtracting the original parameter was 0.07, and the explained variance (R-squared) 80%. The correlation between the original parameter values and the estimated parameter values for η was 0.91.

In simultaneous parameter recovery we observe that the estimates of a and η are less accurate than those of θ_0 and η. This was observed in a 23 rules, 15 students, and 8 instances per rule simulation, with the same distribution of the parameters as in the previous simulations. The results of the simultaneous parameter recovery can be seen in Figure 28. Again we observe that the estimates of a are the least reliable. Also η estimates have a higher variation around the original parameter than those for θ_0 and b. Another

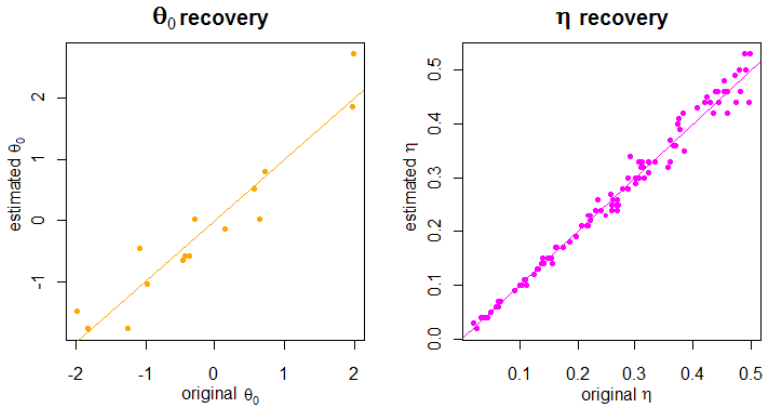


Figure 27: The recovery of the student parameters in a 23 rules, 15 students and 8 instances per rule. For the items b and a were both uniformly distributed. b between -3 and 3 and a between 0.8 and 1.5. $\theta_{s,0}$ had a uniform distribution between -2 and 2, and η_s between 0.02 and 0.50. The lines are no fits, but lines through the origin with slope 1.

pattern seems to be that there is a slight negative correlation between the estimates of student parameters θ_0 and η, of -0.14 (for the concatenated results of all three simulations). Note that the number of instances was only 8 per rule. However, as we increase the instances per rules, this negative correlation persists, and even gets stronger. At 16 instances the negative correlation seems to get even stronger -0.36. The estimates of the η however seem to be better, as shown in Figure 29, where we increased the number of instances per rule several times. The disagreement rate - the proportion of the outcomes where the original and estimated parameters disagree on prediction - is on average 4.6% for the three 8 instances simulations, and 2.6% for the three 16 instances simulations, over all instances. For the first 8 instances only, of the 16 instances simulations, the disagreement rate was slightly higher, 3.1%. This is only one forth higher of the difference between the 8 instances simulations and all of the 16 instances of the 16 instances simulations.

This model, where the student parameters start competence (θ_0) and learning speed (η), depend only on the student and not on the rule seems suitable as a student model. We call this model *extended IRT*. We will now inspect how much accuracy can be achieved by simulation.

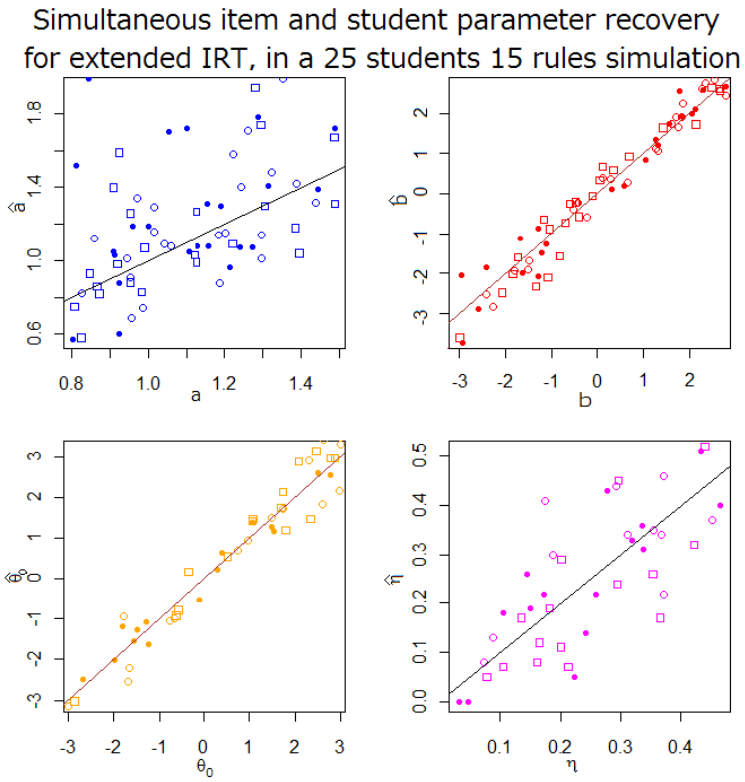


Figure 28: The simultaneous parameter recovery of the student parameters ($\theta_{s,0}$ and η_s) and item/rule parameters (a_r and b_r) in a 23 rules, 15 students and 8 instances per rule simulation. The items b and a were both uniformly distributed: b between -3 and 3 and a between 0.8 and 1.5. $\theta_{s,0}$ is uniformly distributed between -2 and 2, and η_s between 0.02 and 0.50. The lines are no fits, but lines through the origin with slope 1. The squares, open circles and closed circles represent three independent simulations with the same generating distributions for the original parameters. This was done in order to make the results easily distinguishable, and check consistency as there is a limited amount of data points in one simulation.

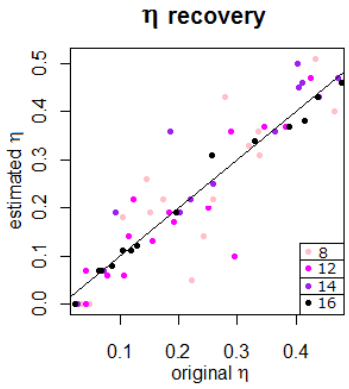


Figure 29: The η_s estimates against the original η_s from a simultaneous parameter recovery of the student parameters ($\theta_{s,0}$ and η_s) and item/rule parameters (a_r and b_r) in 23 rules, 15 students and 8, 12, 14 and 16 instances per rule. b and a were both uniformly distributed. b between -3 and 3 and a between 0.8 and 1.5. $\theta_{s,0}$ is uniformly distributed between -2 and 2, and η_s between 0.02 and 0.50.

4.17 Prediction using extended IRT

Simultaneous rule and student parameter recovery gives us the rule difficulties and discriminativity, and the distribution of the student parameters of an initial group of students. The main purpose of simultaneous parameter recovery however is to get the rule parameters (a and b), so that these can be used when new students are working in the e-tutoring environment. In the e-tutoring environment, the student solves exercises by applying a sequence of rules. Each application of a rule is seen as an instance of the same item (in the vocabulary of standard IRT). The rule in this context has a discriminativity and difficulty. The student parameters η and θ_0 will be updated after each interaction, where the student (succesful or not) tries to apply a certain rule. We say the outcome is negative (0), when the first attempt at the rule fails or the student asks for a hint. This is because after a failed attempt the student receives information from the e-tutoring environment about the step, and is thus not finding the correct step on his/her own. The assumption is that the student learns from each situation a rule can be applied, and his/her competence increases from each application of a rule (whether the outcome was positive (1) or negative (0)). In the previous section we have seen that the model we use for this is the model where the

start competence for each rule is the same, and the increase in competence per instance of a rule is the same for all the rules, for each student.

When applied to an e-tutoring system this model will be used as follows:

- First a training group of students is used to determine the item parameters with. These item parameters are obtained from simultaneous item and student parameter recovery.
- A teacher will look at the item parameters, and set a teaching strategy for the system.
- New student will work in the e-tutoring system, with the existing item parameters. Their competence is estimated using their outcomes. Their outcomes are independent of the outcomes of other students, given the item parameters. This means that their competence and learning speed can be estimated efficiently.
- The teaching strategy may select suitable exercises based on the competence estimates. (Other options are to provide different feedback.)
- Optionally, the item parameters may be re-evaluated after several student have worked with the system. It should then however be ensured that the measurement happens the say way as for the training group of students.

To understand the behaviour of the parameter estimation we investivate the working of the underlying algorithms. First it is important to note that given that there are enough outcomes to base the estimates on, there is an optimum for the student parameters. We can prove this by calculating the hessian matrix for the cross entropy error function in the loglikelihood, and showing that this matrix is positive definite. A positive definite matrix for the cross entropy function means that the cross entropy function is concave. First we define the cross entropy function:

$$\begin{aligned} E(\theta,\eta) = \quad & -\sum_n \sum_i t_{n,i} \ln\left(\sigma(a_n(\theta + d_{n,i}\eta - b_n))\right) + \\ & (1 - t_{n,i}) \ln\left(1 - \sigma(a_n(\theta + d_{n,i}\eta - b_n))\right) \end{aligned} \tag{15}$$

Then we calculate the first derivatives.

$$\begin{aligned} \frac{dE}{d\theta} = \quad & -\sum_n \sum_i a_n(t_{n,i} - \sigma(a_n(\theta + d_{n,i}\eta - b_n))) \\ \frac{dE}{d\eta} = \quad & -\sum_n \sum_i a_n d_{n,i}(t_{n,i} - \sigma(a_n(\theta + d_{n,i}\eta - b_n))) \end{aligned} \tag{16}$$

The Hessian matrix (the matrix of second deratives then becomes):

$$H = \begin{pmatrix} \sum_n \sum_i a_n^2 y_{n,i}(1-y_{n,i}) & \sum_n \sum_i a_n^2 d_{n,i} y_{n,i}(1-y_{n,i}) \\ \sum_n \sum_i a_n^2 d_{n,i} y_{n,i}(1-y_{n,i}) & \sum_n \sum_i a_n^2 d_{n,i}^2 y_{n,i}(1-y_{n,i}) \end{pmatrix} \quad (17)$$

where $\sigma(a_n(\theta + d_{n,i}\eta - b_n))$ has been abbreviated to $y_{n,i}$. We then calculate the product of the Hessian matrix with an arbitrary vector $\vec{u} = (p,q)^T$:

$$\begin{aligned} \vec{u}^T H \vec{u} = \sum_n \sum_i p^2 a_n^2 y_{n,i}(1-y_{n,i}) \quad & + p\, q\, a_n^2\, d_{n,i} y_{n,i}(1-y_{n,i}) + \\ & q^2 a_n^2 d_{n,i}^2 y_{n,i}(1-y_{n,i}) \end{aligned} \quad (18)$$

When we rewrite this formula we obtain the following formula:

$$\vec{u}^T H \vec{u} = \sum_n \sum_i a_n^2 y_{n,i}(1-y_{n,i})(p + q d_{n,i})^2 \quad (19)$$

We know that this sum is always positive as a_n^2 and $(p+qd_{n,i})^2$ are squares and $y_{n,i}$ is a probability and therefore between 0 and 1. We thus have proven that the Hessian matrix is positive definite.

Secondly, for a limited number of outcomes, the value of η can be irrelevant, also in terms of the loglikelihood/cross entropy. Therefore it is a design choice in the algorithm, to prefer the value 0 in such cases, as when the parameter is not relevant. The value 0 has the effect that η does not seem to be there at all.

The implementation of the discrete maximum likelihood optimization, has a specific order in which to test the loglikelihood, where low η is preferred. `etaMin` is usually 0; no learning whatsoever:

```
boolean mlDiscreteStudents(
      thetaMin, thetaMax, stepSizeTheta,
      etaMin, etaMax, stepSizeEta)
{
         foreach(student s with his/her outcomes){
                currentBestLL  = NegativeInfinity
                currentStudent = this.students.get(s)
                etaBefore      = student.learningParameter
                theta0Before   = student.startCompetence
                for(th=thetaMin; th<=thetaMax; th+=stepSizeTheta)
                    for(e=etaMin; e<=etaMax; e+=stepSizeEta)
```

```
                        testLL= loglikelihood(student.outcomes,
                                 as, bs, e, th);
                        if(testLL>currBestLL) then
                                currBestLL=testLL
                                currBestTh=th
                                currBestE =e
                        endif
                    endfor
                endfor
                if(etaBefore <> currBestE OR theta0Before<>currBestTh) then
                    changed=true
                endif
                student.learningParameter=currBestE
                student.startCompetence
        endforeach
        return changed
}
```

The simulations use this function in prediction as follows: (a) the (simulated) student is given a start competence and a learning parameter, (b) for each rule 1 random outcome is generated. Disregarding the original student parameters, an estimate is made of the student parameters (using the above function). A random rule number is selected and an outcome is generated on the basis of the rule parameters and the original student parameters. The random outcome is compared to the prediction, and the number of correct predictions and total predictions are updated. Then the outcome is added to the data, and a new estimate is made for the student parameters. Then a random rule number drawn again, and so on until the total number of outcomes reaches a certain number (which is a program parameter). Using this procedure we can see what happens to the estimates of the student parameters in a sequence (Figure 30). We observe that the difference in η_s and $\theta_{s,0}$ between outcome number t and $t-1$ seem to be negatively correlated. Furthermore we see that the estimates stabilize as the amount of evidence increases, and also come closer to the original parameters. In Figure 30 the item parameters used to generate the outcomes, and to estimate the parameters with, are the same. In practical use however, this will not be the case - as the rule parameters will be estimates themselves. In order to show the difference we estimated the student parameters by simulation again. This time the rule parameters were taken from a simultaneous student and rule parameter recovery simulation (8 instances per rule), and the rule param-

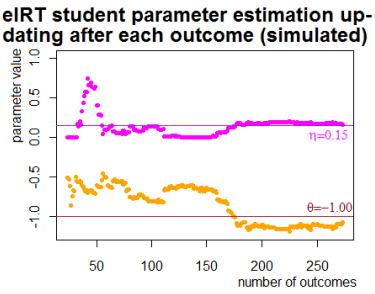


Figure 30: The η_s and $\theta_{s,0}$ estimated by discrete maximum likelihood optimization as the outcomes (instances of a rule) come in one by one, in sequence. The "true" values of the student parameters are shown as horizontal lines.

eters to generate the outcomes with were the original parameters from the same simulation. The accuracy of the first was 84.4%, the accuracy of the prediction simulation using estimates of rule parameters was 83.6%. This is not a significant difference. Apparently the rule estimated from the recovery were sufficient to make proper student parameter estimates with, for new students.

When we look at the accuracy of the predictions we obtain the following tables: Table 2 is the accuracy obtained using estimated item parameters from a simultaneous parameter recovery, Table 3 is the accuracy obtained using the original item parameters. The latter set seems to be slightly higher, as expected. This difference is not significant. When a two-sample t-test is run, the p-value for the null hypothesis (equal means) is 0.58. A difference in means that is significant is the difference between the last two rows of both tables and the rest, the null hypothesis "true difference in means is equal to 0" has a p-value of 0.006. Outcomes for students with η values of 0.45 and 0.5 can be predicted better for 250 intances, than students with $\eta \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.35, 0.4\}$. This can be explained by the fact that these students go through the probability region with the highest uncertainty quicker.

It should also be noted that the predictions using extended IRT (eIRT), for the 250 instances of random rules, after one outcome was known for all rules for each student, always yielded a better or equal accuracy than just predicting the majority class (the outcome that had occurred most up till then).

The accuracy for eIRT was between 83% and 84% on average. This

$\eta \downarrow$ / $\theta_0 \rightarrow$	-3	-2	-1	0	1	average
0.05	214	200	219	212	201	209.2
0.1	210	206	218	205	207	209.2
0.15	199	206	209	203	203	204
0.2	206	212	212	201	210	208.2
0.25	209	207	198	195	222	206.2
0.3	209	207	196	201	223	207.2
0.35	198	199	212	200	216	205
0.4	196	198	209	208	221	206.4
0.45	212	211	206	218	209	211.2
0.5	203	215	205	226	227	215.2
average	205.6	206.1	208.4	206.9	213.9	208.18

Table 2: The counts of accurate predictions in a simulation where all rules are given one outcome to start with (as initialization) and then get 250 instances of rules, randomly spread over all rules. Before each rule a prediction is made by maximum likelihood estimation (IDLLO), and the accuracy is counted. For each pair of θ_0 and η the results are presented in this table. For the item/rule parameters, the recovered parameters where used as presented in Figure 28.

makes it a useful model for prediction. To test whether the model is still useful for fewer outcomes we run the same simulation with only 100 random outcomes, instead of 250. For $\theta_0 = -1$ and $\eta = 0.25$, using the original parameters the result was 80% using the original item parameters (versus 87% for 250 outcomes), and 79% versus 79% for when the estimated item parameters were used. The accuracies were lower for only 100 outcomes (e.g.76% for $\theta_0 = -3$ and $\eta = 0.25$, using the estimated parameters) but still useful for prediction.

4.18 Summary of the student model creation

We extended the 2PL IRT model for usage in rule based e-tutoring systems. We have designed a model, where the competence of a student depends on a start competence, and the number of times a student encounters a certain rule. We have shown, that for the model to be usable as a descriptive model, the assumption that the start competence and learning parameter depend only on the student (not on the rule), is necessary. Not having this assumption leads to unacceptably high variance in the start competence. The probability model we propose is given in equation 14. We call this

$\eta \downarrow$ / $\theta_0 \rightarrow$	-3	-2	-1	0	1	average
0.05	214	200	228	209	187	207.6
0.1	211	205	211	192	196	203
0.15	208	196	207	217	193	204.2
0.2	216	203	206	207	215	209.4
0.25	214	203	219	205	218	211.8
0.3	206	205	206	215	203	207
0.35	199	201	202	213	223	207.6
0.4	203	206	214	212	225	212
0.45	207	204	216	215	222	212.8
0.5	211	207	205	223	234	216
average	208.9	203.0	211.4	210.8	211.6	209.14

Table 3: The counts of accurate predictions in a simulation where all rules are given one outcome to start with (as initialization) and then get 250 instances of rules, randomly spread over all rules. Before each rule a prediction is made by maximum likelihood estimation (IDLLO), and the accuracy is counted. For each pair of θ_0 and η the results are presented in this table. For the item/rule parameters, the original parameters from the simulation in Figure 28 were used.

model *extended IRT*.

We have shown by simulation that this model is a reliable model in terms of parameter recovery. We have done this by taking the original parameter as a "model" for the student parameters, and observe that this accounts for 99% of the variace (R-squared statistic) for both $\theta s, 0$ and η_s in a 100 students, 100 rules and 50 instances per rule simulataneous parameter recovery simulation. For a smaller simulation (23 rules, 15 students and 8 instances per rule) the explained variance was still 91% for θ_0. For η we find that the variances was high, but drops quickly as we increase the number of instances per rule (Figure 29).

Finally, we have shown that the model is usable as a predictive model as well. Using the recovered item parameters and new students with set student parameters, an average accuracy of 83.6% could be reached, as shown in Table 2.

With the extended IRT model we have a model in hand with which we can attempt to describe and explain data we gather from intelligent tutoring systems based on rules. We do however not yet know how well this model works in practice. To evaluate this we will have to look at the accuracy

of prediction, and evaluating the parameter estimates with domain experts. The gathering of the data will be explained in chapter 5. After which the data will be analyzed by hand in chapter 6, and then by using our extension to IRT in chapter 7.

5 Data acquisition and transformation

In this chapter we discuss how we obtain and process the data. We discuss the form of the data, the various transformations we perform and the analyses.

5.1 Test student population

As mentioned in the introduction, the first thing to consider while setting up an experiment using IRT, and experiments containing competence in general is the student population. We tried to find a suitable test population by considering different groups of students.

The first test run with this domain was performed using another master student as a test subject. However, quickly it appeared that all the errors he made where lapses in concentration, or miss-typing. Therefore the mistakes are uncorrelated to his understanding of the subject. His competence was so high, that the exercises posed no challenge whatsoever. The experiment was also performed with an undergraduate student who had completed a course in formal logic and was in his third year of the bachelor computer science at Utrecht University. The results were similar results however. A master student in education, who had a bachelor in physics, but had next to no prior knowledge of the logic domain could not solve the exercises after just the experimental instruction and reported more time to comprehend the domain was required, and offered to repeat the experiment after a week or so, so there would be time to read about logic.

We decided to use undergraduate students that did not yet complete a course in formal logic. These students can either be HBO or WO students. We also concluded that an understanding of the basic concepts of booleans/logic was required. Students who follow ICT related programs soon obtain programming experience (and/or electronic signal processing) and therefore are not unfamiliar with boolean expressions. Formal logic and rewrite rules can thus soon be comprehended by this group. HBO and WO students without training in formal logic have thus been included in the population. This resulted in a population of students which make mistakes, but does not despair due to unfamiliarity with the domain.

The total number of students was fifteen. The applied science students were first and second year students in Business Informatics and Technical Computer Science. The students at Utrecht University had Computer Science as a major and were all first year students. An important difference between the two groups is that the latter group was taking a logic and set

theory course at the time, as a part of which the disjunctive normal form was discussed in class.

The applied science students got a special thirty minutes instruction about the experiment. The first year Business Informatics students just completed a programming course, and a course in computer systems including electronic signal processing. They received one extra one hour lecture in formal logic and the thirty minutes instruction. The second year technical computer science student had quite a lot more programming experience, and did not receive this one hour lecture in formal logic, but one brief (10 minutes) extra instruction on formal logic notation and the thirty minutes instruction. Both groups received a thirty minutes instruction on the disjunctive normal form. Even though prior knowledge was acquired in a different manner, care was taken that the science and applied science bachelor student were provided with roughly the same information.

Besides lectures and oral instruction, all subjects received and were requested to read experimental instructions on paper, after which they could ask questions. These instructions will be made available upon request.

5.2 Acquisition

The data was acquired using the a locally installed version[28] of the `ideas` logic environment using the `GenExas` interface, on the XAMPP or XAMPP lite webservers. Four sessions were held, two at Utrecht University of Applied Sciences[29], with 5 and 6 students, and two at Utrecht University with 1 and 3 students. The students installed the environment on their own laptop, using a CD provided by us. The CD contained a windows installer for XAMPP, the `GenExas` environment including `ideas` in the form of a cgi file, installation instructions and the experimental instructions in the form of a pdf file. The operating systems the students used were: windows XP, windows vista, windows 7 and mac OS using an XP virtual machine.

The sessions all took place in the 2010-2011 academic year, in the second half of the fall semester (in the Netherlands). The author of this thesis organized and was present during the sessions. No disturbances occurred during the sessions. The subject students were not allowed to talk to each other. The students were instructed to finish the 25 exercises step by step, and in silence. Although silence was not preserved the entire time, none of the students reported that they were unable to concentrate. Two subjects reported

[28]The branch URL is: `https://ideas.cs.uu.nl/svn/Feedback/branches/diederik`

[29]The name "Utrecht University of Applied Sciences" is an unofficial translation, and is officially "Hogeschool Utrecht" in Dutch.

that due to the length of the session, the capability to focus decreased during the last few exercises. In order to create a relaxed atmosphere drinks (non-alcoholic) and snacks were provided. At the Utrecht University of Applied Sciences the students who had finished waited for each other to finish in silence, and at Utrecht University the students who were done e-mailed the results and left in silence.

At several points during the sessions the system appeared to malfunction. It did not present a hint when it should. In those cases we provided a hint ourselves, in the form of the name of the applicable rule, as the system should have done. Upon request of the subject, sometimes an auto-step was executed by the experimentor present as well. Care was taken to ensure that the right button in the system was pressed as well.

After the sessions the subject students sent in their log files by e-mail. The log files were initially stored locally.

The sessions at Utrecht University of Applied Sciences lasted longer (both approximately 3.25 hours), than the ones at Utrecht University (approximately 2, and 2.5 hours). This can probably be explained by the difference in experience with the logic domain. The Utrecht University group probably has a different starting competence from the Utrecht University of Applied Sciences group.

The students we talked to after the sessions (all except one at the university of applied sciences and half at Utrecht University) reported they found the experiment instructive and useful. Three students reported fatigue. One student misinterpreted the experimental instructions (he thought he only ought to do one step for each exercise) and restarted after he had noticed this - this however resulted in that he had seen the first step of the exercises already, so that his short term memory provided most of the answers there. We included his first attemps at all the first steps, but chose to ignore the rest of the data for this student, in order to ensure equal conditions for all test persons.

5.3 Transformation

The raw data obtained from the experiments are JSON format service requests and responses. The requests are generated when a student presses a button.[30] The possible services are: generate (which will generate a new exercise, which is replaced by an exercise on the predefined exercise list),

[30]Most buttons in the interface generate a request. The ones that do not are the logic keyboard input buttons, the back button, and the rewrite and about buttons. We provide a screenshot of the interface (GenExas) in appendix A.

submit text (which says that something has been edited, and asks `ideas` for feedback), ready (which says that the student thinks that the current form of the logic formula is in DNF, and requests `ideas` to check whether this is true), hint, step, and auto-step. A hint provides the student with the name of the rule he/she should apply when following strategy `dnfStrategy2`. A step provides the student with the intermediate solution which results from applying that same rule, in the feedback text field. Auto-step performs the rule on the intermediate solution in the working area. (See appendix A for a screenshot of the interface for clarification.)

For the purpose of our experiments, the relevant information in the request is:

- which student made the request
- which rule the student tried to apply
- whether this attempt was succesful.

In Figure 31 two example requests are given. From these requests the important information is that the student with id 1, has succeeded in performing implication elemination, and then, correctly recognised that the formula was in DNF. We translate this information to 3-tuples of the form $(studentID, ruleNr, outcome)$. The rule number ($ruleNr$), is the rule we assign the attempt to. This is either one of the rules available in the domain, or pressing the ready button. The numbers used as an ID for a rule are given in Table 4. Assigning a step made by a student to a rule is trivial when the step is a correct application one of the known rules, or when the student presses the ready button. In these case `ideas` always succeeds in recognizing the step for us. In the case of an incorrect step `ideas` can identify a "buggy rule", which is a common mistake. In that case, the buggy rule is linked to a correct rule, and the rule number is still easily assigned. In all other cases the experimentor assigned rule IDs manually.

At each intermediate solution one or more rules can be applied. After a rule is correctly applied, a new intermediate solution arises. Often, a student can make several mistakes before a correct rule is applied. Mistakes, or asking for help (by means of the help button) will lead to aid from the system. If aid is supplied prior to the correct application of a rule, we classify this step as wrong. Several attempts at the same rule is classified as one wrong step. To show how this classification of attempts works, we present some examples from the experimental data.

First, consider the following logical formula: "$(q \wedge r) \vee (r \wedge r)$", which the subject student (with ID 1) first rewrites into "$(q \wedge r) \vee T$". After receiving

ruleNr	rule name	example
1	Commutativity and	$p \wedge q \implies q \wedge p$
2	Commutativity or	$p \vee q \implies q \vee p$
3	Distribution and over or	$(p \vee q) \wedge r \implies (p \wedge r) \vee (q \wedge r)$
4	Distribution or over and	$(p \wedge q) \vee r \implies (p \vee r) \wedge (q \vee r)$
5	Idempotency and	$q \wedge q \implies q$
6	Idempotency or	$q \vee q \implies q$
7	Tautology or	$q \vee \neg q \implies T$
8	Contradition and	$q \wedge \neg q \implies F$
9	And true	$q \wedge T \implies q$
10	Or true	$q \vee T \implies T$
11	And false	$q \wedge F \implies F$
12	Or false	$q \vee F \implies q$
13	Neg true	$\neg T \implies F$
14	Neg false	$\neg F \implies T$
15	Double neg	$\neg\neg p \implies p$
16	DeMorgan on or	$\neg(p \vee q) \implies \neg p \wedge \neg q$
17	DeMorgan on and	$\neg(p \wedge q) \implies \neg p \vee \neg q$
18	Implication elemination	$p \rightarrow q \implies \neg p \vee q$
19	Equivalence elemination	$p \leftrightarrow q \implies (p \wedge q) \vee (\neg p \wedge \neg q)$
20	Absorption outer or	$(p \wedge q) \vee q \implies q$
21	Absorption outer and	$(p \vee q) \wedge q \implies q$
22	Tautology F impl	$F \rightarrow p \implies T$
23	Tautology A impl A	$p \rightarrow p \implies T$
24	Ready?	$q \vee r \vee p = ready? \implies yes$

Table 4: The rule with the numbers used to represent them during data analysis. The order of the columns is: rule identifier, textual description and example.

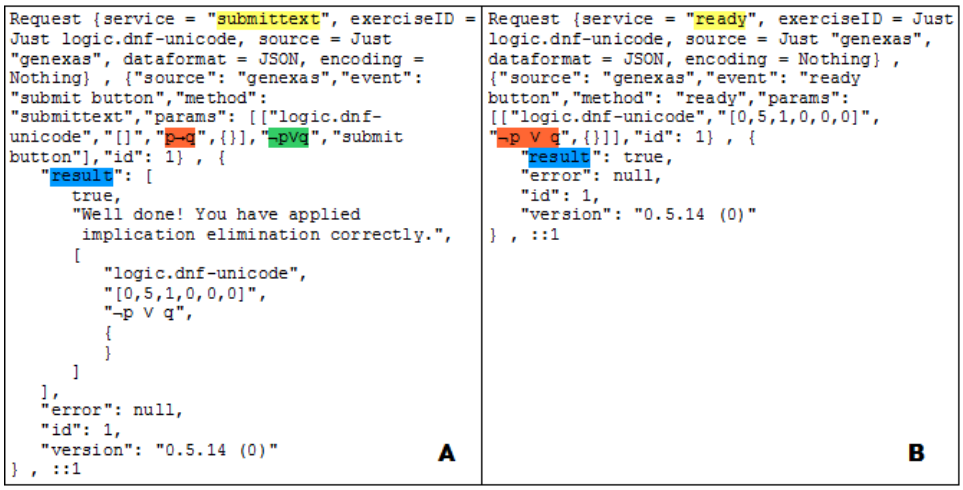


Figure 31: Two requests by a student with id 1, one submit-request and one ready request. The yellow highlight shows the request. The orange highlight shows the formula before the edit action of the student, the green after editing. Note that for the ready request, the expression has not been edited. After the result tag, highlighted in blue, it is shown how `ideas` replies.

feedback ("*You rewrote $r \wedge r$ into T. This is incorrect. Did you think that phi AND phi is equivalent to True? This is not the case. Idempotency of AND means that phi AND phi is equivalent to phi. Press the Back button and try again.*") the student writes "$(q \wedge r) \vee r$". This leads to only one tuple in the log file: $(1, 5, 0)$ (student ID 1 tries idempotency of "and", and failed).

On the other hand "$\neg\neg p \vee \neg\neg q$", "$p \vee \neg\neg q$", "$p \vee q$" would lead to two tuples: $(1, 15, 1)$ and $(1, 15, 1)$ (student ID 1 tries double negation, and succeeds, two times). This example brings us to another case which we chose to handle in a certain way. What if the student writes "$\neg\neg p \vee \neg\neg q$", "$p \vee q$"? This would lead to a message saying that the student has "*combined multiple steps, or made a mistake*". The student then realizes it is two steps, and does one double negation rewrite at a time. We chose to not count this as a mistake. The reason for doing so is that this sequence represents a misconception about the system, rather than a conceptual mistake. We want to measure competence of the student *in the domain*, and therefore

neglect mistakes that are purely due to operations of the system.[31]

Sometimes we find errors to be difficult to classify. We get the impression that the student takes "a wild guess", or is trying to combine different aspects of his/her knowledge into creative rules. The system always fails on this, as `ideas` implements known rules. We try to attribute each mistake to a rule, but when we cannot do this we write an unidentified mistake. These mistakes are ignored for data analysis because we cannot estimate competence for the rules. One argument to justify this, is that wild guessing does not display ability, and does not help the student gain more knowledge (the system only says that he/she is wrong). The second argument is that if it is a clever combination of rules, the student can then sequence this into steps, as mentioned in the previous paragraph.

The student has the option to go back to a previous point in the solution. Students are known to do this[32] in order to redo a situation they failed and so remember it better, when they suspect themselves of having taken a detour, or by mistake. This often results in taking steps the student has already done before. We ignore this data in the log file. The reason we do this is that the correctness of the steps comes from the (short-term) memory of a specific situation rather than competence of the rule in general.

Note that for each exercise, ready is only included in the final results once. So pressing ready in a wrong situation, implies that for that exercise, only one tuple with the ready step will appear in the log file after the transformation (with outcome '0').

After the transformation, for each student, a sequence of tuples is obtained. The first ten tuples for student 1 for example are: $(1, 17, 0)$, $(1, 15, 0)$, $(1, 3, 0)$, $(1, 5, 0)$, $(1, 24, 1)$, $(1, 5, 1)$, $(1, 18, 0)$, $(1, 13, 1)$, $(1, 11, 1)$, $(1, 18, 0)$. This sequence contains both information about which steps were attempted, in which order, and information about the correctness of these attemps.

The series of tuples can be transformed into a sequence of outcomes per step. As an example of which, these sequences for student 1 are shown in Table 5.

[31] We do this for all combinations of two or three steps, as long as the student then displays the ability to do the sequence step by step.

[32] We asked.

1	1 1 1 1 1
2	1 1 1
3	0 0 0 0 0 1 0 1 1 1 0 1 1 1 1 0 1 0
4	0 0 0 0
5	0 1 0 1 1 0 1 1
6	0 1 1 1 1
7	1 0 0 0
8	1 1 0 1
9	0 0 0 1 1 1
10	0 1
11	1 1 1 1 1
12	1 1 1 0 1 1 1 1
13	1 1 1 1
14	
15	0 1 1 1 1 1 1 1 1 1 1
16	1 1 1 0 1 1 1 1 1 1 1 1
17	0 0 0 0 0 1 1 1
18	0 0 0 0 1 0 1 0 0 0 0 1 1 1
19	0 0 0 1 0 1 1 0
20	0 1 1 0 1 0
21	0 1
22	0
23	1
24	1 0 1 1 1 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0

Table 5: The outcomes of all attempts of student 1 shown as a sequence per rule.

6 Interviews and manual inspection of the data

In this chapter the data will be inspected and analysed without the extended IRT model we created. We do this on the basis of visual inspection of different views of the data and data aggregation. We also consult domain experts to determine the order of the difficulties of the rules and the start competence and learning speed of the students. We also ask them to consider the order of the overall or average competence of the students. These results will later be compared to the results learned from data using extended IRT. The purpose of this chapter is to describe and partially explain the data, in order to form useful descriptions for what is happening. We use this as input for evaluating the parameter learning we do in chapter 7.

All student names are withheld for privacy reasons. Fake names are used instead.

We use three different views of the data. Firstly there is the raw data (V0), which is a sequence of tuples of studentID, ruleID, and outcome. Secondly we have the view where the data is ordered per student, as sequences of outcomes (zeroes and ones) per rule (V1). V1 is useful in order to view the progress a student makes by hand, and to learn their competence from data automatically. Views V0 and V1 are presented in appendix B. Thirdly there is the rule view (V2), which shows per rule, all the sequences of outcomes for that rule per student. View V2 is presented in appendix C. Some aggregations of the data are provided in the text.

6.1 Comparing students and rules by manual inspection

The data of all students is provided in appendix B, in the form of tables, ordered per rule (V1). Under these tables we display the data in the form of a sequence of tuples "(studentID, ruleID, outcome)" (V0). This sequence is in the same order as the order in which students entered the data. A second view (V2) is provided in appendix C, which shows tables of all the outcomes for a rule, with one line of outcomes (in sequence) per student. First we will make some general observations, then we will compare students, and finally we will compare rules.

The first thing that draws our attention is the fact that a large majority of sequences per rule has more negative outcomes "0" at the beginning of the sequences than at the end. This supports our theory that learning is happening and that the competence only increases by working in the e-tutoring system.[33] Typical examples of this are e.g. students one and two

[33] Of course, we cannot observe competence and the corresponding probabilities directly.

Tables 11 and 12).

We observe also that not everyone has an equal number of outcomes for each rule. This is partially explained by detours, and personal preference. An example of different steps with the same result is: "$\neg(p \vee p)$", which can be written into "$\neg p$" by idempotency of the or, but also by DeMorgan and then idempotency of the and. Part of this can also be explained by students giving up on exercises, and using rules that do not lead to better solutions (which is usually true for rule 4, distribution of $\vee$ over $\wedge$). Having more or fewer outcomes per rule does not seem to indicate higher or lower competence for these rules (compare students 3, 5 and 6 in Tables 13, 15 and 16).

One rule is used that is not in the list. This rule is an alternative to standard equivalence elimination: "$p \leftrightarrow q \implies (p \rightarrow q) \wedge (q \rightarrow p)$". It has been used just once by one student, and we therefore exclude this rule from further (automatic) analysis, because no difficulty or discriminativity can possibly be determined for it. Rules 1 and 2 (the commutativity rules) are always applied correctly by all students. We exclude these rules for the same reason.

6.1.1 Comparing students

First we compare the three different groups of students:

- technical computer science students at HU (students 1 to 5)
- computer science students who were taking the logic course at UU (students 6 to 9)
- business informatics students at HU (students 10 to 14)

The number of positive outcomes, the total number of outcomes and the proportion of positive outcomes are given in Table 6. The business informatics group seems to contain most of the students with a less than average proportion of correct outcomes. Two of these students however, have an exceptionally low number of outcomes in total, and are thus only showing the beginning of the learning curve. We could say the Utrecht University computer science students got the highest proportion of outcomes right on average (which would be expected as they have had the most instruction in the logic domain). None of the groups however dominates another in terms of proportion of positive outcomes - the lowest one of the group with the higest average does not have a higher proportion of positive outcomes than the highest of the two other groups.

student	# correct	total no. of outcomes	proportion correct
1	97	162	0.599
2	104	157	0.662
3	161	211	0.763
4	169	196	0.862
5	45	140	0.321
6	165	178	0.927
7	140	165	0.848
8	149	181	0.823
9	130	176	0.739
10	15	30	0.500
11	141	199	0.709
12	66	149	0.443
13	117	153	0.765
14	14	52	0.269

Table 6: The number of positive outcomes, the total number of outcomes and the proportion of positive outcomes per student.

When we compare the student outcome tables we make several observations. Even though on the whole the outcomes of two students can be comparable, there may be one or two rules which show really different results. For example, when we take the Utrecht University students (Tables 16 - 19) we observe that student 6 seems to have most answers right, and student 9 least. Both students 6 and 9 have most negative outcomes for the absorption rules (rules 20 and 21). Students 7 and 8 are in between of students 6 and 9 in terms of the proportion of negative outcomes. Student 7 uses absorption less than the other students and has most outcomes wrong - which need not indicate anything as this student might not have taken the opportunity to learn these rules. The outlier seems to be student 8, who has all outcomes but one in the absorption rules correct and used the rules slightly more often as well. When it comes to DeMorgan's laws however, student 8 has a much lower correctness rate than students 6 and 7 but comparable to 9. This might indicate the possibility that a student may have a favourite rule - which he/she is exceptionally good at and likes to use - compared to the other rules.

There are some observations we make which seem to support our student model. Students who make mistakes intertwined with correct answers a lot, do so over all rules (e.g. student 12 in Table 22), while other students

display an all negative, an intertwined and an all positive region (student 1 in Table 11). There are also students which show an intertwined and a positive region only (student 6, Table 16) or mainly the negative and intertwined regions (student 5, Table 15). There is also one student we left out because he did not get any answer right. These examples would all seem to correspond to the pattern where a low, a competence around the difficulty, and a high competence are achieved in that order (not necessarily including all parts). This pattern would arise when the probability of a correct outcome is described by 2PL IRT, and the competence steadily rises - as in our student model.

In the sequences of tuples (V0) we observe that at the end of the sequences there are more positive outcomes than in the beginning. There are two possible explanations for this: the domain knowledge increases, and the knowledge about the individual rules increases. The latter is described by our proposed extended IRT model. These two explanations are however not mutually exclusive. When we look at the data more closely however, we observe that the number of times a rule has been performed is the dominant concept.[34]

To summarize, inspecting the outcomes per student manually, the data largely corresponds to our proposed exended IRT model. The only observation that does not seem to fit, is the phenomenon that some students seem to have a "favourite" rule. This might be an imperfection in the model.

6.1.2 Comparing the rules

Now let us move to the analysis of the outcomes from the perspective of rules. The sequences of outcomes per rule are provided in appendix C.

In Table 7 we show the proportions of positive outcomes per rule. Rules 1 and 2 (the commutativity rules), and 14 ($\neg F \implies T$) have only positive outcomes and are thus too easy for the students. These rules should thus be excluded from automatic analysis as their difficulty would be negative infinity and the discriminativity undeterminable.

How well do the proportions of positive outcomes correspond to the difficulty of an outcome? There ought to be a negative correlation between the two. However, the model we propose also includes discriminativity, which raises the probability of a positive outcome at low competences and decreases

[34]This can also be observed from Table 10 in the next chapter, where the number of positive outcomes in the end of the sequence is not that much more, once you exclude the easiest rules, and the "ready-rule". We did try out the alternative hypothesis anyway, but that returned significantly worse results.

rule	# correct	total no. of outcomes	proportion correct
1	40	40	1.000
2	15	15	1.000
3	171	263	0.650
4	42	55	0.764
5	77	101	0.762
6	80	91	0.879
7	40	63	0.635
8	52	67	0.776
9	53	72	0.736
10	14	33	0.424
11	36	41	0.878
12	100	115	0.870
13	42	46	0.913
14	10	10	1.000
15	148	164	0.902
16	104	151	0.689
17	77	116	0.664
18	142	199	0.714
19	57	94	0.606
20	24	56	0.4295
21	20	55	0.364
22	2	6	0.333
23	2	5	0.400
24	164	290	0.566

Table 7: The number of positive outcomes, the total number of outcomes and the proportion of positive outcomes per rule.

the probability of a correct outcome for students with a high competence. We think discriminativity is a useful concept, as for some rules there are more mistakes later in the sequence, and more positive outcomes in the beginning, than for other rules. An example can be found in the distribution of $\wedge$ over $\vee$ (Table 25) compared to DeMorgan's laws (Tables 38 and 39).

Rule 24, the student pressing the ready button, gives rather strange results. Table 46 shows that there does not really seem to be much of a learning effect. (The exception to this seems to be student 14.) The reason for this is that many students did not take the ready button very seriously (they pressed the button just to check whether they were done, before properly checking themselves). Also the system sometimes provides the information that the student is done without the student asking for it. For this reason we will try to learn the model with, and without this "rule".

The low proportion of correct answers at the absorption rules (20 and 21) did not surprise us, as we expected these to be difficult. What we did not expect was the low proportion of positive outcomes at rule 10, "or true, is true". When we look at Table 32 we see that not many instances of this rule were performed. Still it is surprising when comparing it to the other true/false rules (rules 9-12). A student mentioned that this rule was logical, but counterintuitive as he/she interpreted it as "*or whatever*" the first time. This misconception may very well persist for a while, even though the student already knows that what he/she is doing wrong.

We consider the possibility of persistent misconceptions. We checked whether students made the same mistakes for the same rule. If the student begins applying a rule and does not yet master this rule, usually many different mistakes are made. Later when the student makes less mistakes, these mistakes are often the same mistakes. A typical example of such a persistent mistake is forgetting to turn $\vee$ into $\wedge$ while applying DeMorgan. We observed several examples of persistent mistakes disappearing slowly. When asking the students about this phenomenon they said they already knew the right way to do it, but the mistake "slipped in" anyway.

We interviewed students to get their impressions of the experiments. We interviewed several students more than one month after participating in the experiment, but without showing them any results. We present a summary of the interview with Alan. We asked Alan about the difficulty and discriminativity of the rules, about learning a strategy and his impressions of the experiments. He reports that when he was doing the experiments, the rules were all rather new to him. Because of that he was concentrating on learning the rules, rather than planning ahead. He says that he started to seriously think about the strategy, at the very end, but mainly after the

experiment. "*The system is good to study rules with. It feels rather like flash cards, which I used to study English vocabulary. This way of studying does not seem very suitable to learn strategy with. Inputting into system has been made so easy that it feels like a game. You just try anything you can think of, to get closer to the end of the "level". He considers his own strategy to have been "persistent trial and error". Some patterns probably have formed themselves in my actions, but it was not consciously.*" Many other students also reported that they were annoyed by, and generally ignored any feedback that said that they were not following the standard strategy. Alan reports that implication elimination was often the first thing he did, as "*whatever you do, you have to get rid of those anyway*". He also said implication elimination was difficult at first, but once he remembered it he did not do it wrong anymore. He contrasts this with the example of DeMorgans laws, where he sometimes made an error, even though he knew the rule very well. He also reports that he already figured out DeMorgan, because of his programming experience, and expects others to have done so as well. Distribution was new to him. When considering difficulty, Alan considers absorption (rule 20 and 21), as definitely the hardest, and rules 1, 2 and 5 to 15 as easier than the rest. In general he thinks that rules with many literals and operators are harder. Alan reports to have been sloppier than he was when doing similar exercises on paper, due to the computer interface, which made the experiment feel like a game to him.

The interview with Alan was a typical result. Most of his impressions correspond to those of the other students we spoke with. The most interesting of the other interviews was that of Edgar and Judea together, who judged the difficulty of the most difficult items differently. They said that rule 19, then rule 18 and then rules 3 and 4 were the most difficult. We looked at their data, whether this difference was also reflected in the sequences they had for these rules. This was not the case: both of them made most mistakes for rules 20 and 21. One of them did not make mistakes for rule 3 and 4 at all, the other made relatively more mistakes for 20 and 21 than for 3 and 4 in the beginning of the sequences for these rules. (The number of outcomes for rules 3 and 4 was much more than for rules 20 and 21). Both of them made only one or two mistakes for rules 18 and 19.

Most students found it very hard to make a statement for discriminativity. Edgar and Judea compiled a discriminativity order for the rules together. The discriminativity from high to low according to them should be rules: 13, 14, 15, 1 and 2 together, together with 5 to 12, then 16 and 17, 20 and 21, 3 and 4, 18, 19, and 22 and 23, according to Judea. Edgar did not agree with this ranking entirely and would place rule 18 before rules 3

and 4 instead.

We checked whether we could find any patterns in strategies that students follow. We could see that there was not much consistency in rule preference in situations. Some students seem to prefer depth first solutions (rewriting one part of the formula until it is close to DNF, and then looking at the rest), while other students solve several parts of the formula simultaneously. A slight preference seems to exist for applying the same rule twice. To make good observations about strategy forming, we need more data from longer, or repeated, experiments. For now the data seems to correspond to the idea that the students used a strategy of persistent trial and error.

6.2 Hypotheses

From the sequences, talking with students and our domain knowledge of both logic and education and the `ideas` system we form hypotheses about the rules and the students. We compare this with the results obtained from automated learning from data in chapter 7.

About the students we expect the UU students to have the highest start competence, as they were doing a 7.5 ects course on formal logic, then the technical computer science students of the HU as they had one and a half years of programming experience, but not much experience in formal logic, and finally the business informatics students at the HU, as they were first year students, with only one programming course and a course in computer systems.

Two students had very few outcomes, only showing the beginning of their learning curve. Due to little data, the results for these students may be not very reliable, especially in terms of the learning parameter. When we look at the data for these students there are very few positive outcomes at the beginning, and very many positive outcomes at the tail of the sequences for the rules they have more outcomes in. This may be caused by random effects, but when run through automated parameter estimation by maximum likelihood estimation, will probably be estimated as a very low start competence and a high learning speed.

From our knowledge of the domain we hypothesize that the absorption rules are the most difficult, as it takes quite some insight before one can understand these rules. The difficulties of the absorption rules should be close together. After which we expect the distribution rules to follow, but as in the experiment distribution of $\vee$ over $\wedge$ does not have to be used, this is only done by students who choose to perform it. Distribution of $\vee$ over $\wedge$ is never hinted by the system, which may lead to a lower difficulty. The next ones

in line are probably equivalence elimination and implication elimination. Equivalence elimination is probably slightly more difficult than implication elimination, and easier to make a mistake with even when you know the answer (which would lead to a lower discriminativity). This hypothesis also corresponds to the remark that rules that contain more operators and literals generally feel more complex.

In the lower regions we expect to see $\neg T \implies F$ and $\neg F \implies T$ and double negation elimination. The latter being probably the harder of the three. Not true and not false are probably the most basic concepts, and not not is yes is quite well known. In fact we have already seen that no mistakes were made with $\neg F \implies T$. Idempotency rules are probably easy as well, followed by the True/False rules.

Rule 24, saying a solution is in DNF by pressing the "ready" button is, as said before, a strange rule, as many student did not take it seriously, and the system may provide hints the students did not ask for. This probably leads to a very low discriminativity.

From the discussion with the student saying that rule 10, $p \vee T \implies T$ is easy to make mistakes with, we suspect that this rule might have a surprisingly high difficulty. Also, a lower discriminativity can be expected if this statement is true.

Rules that have a high discriminativity are rules that you either "get or do not get". We suspect this to be the case for double negation, implication elimination, equivalence elimination and the absorption rules. As explained before rule 4, distribution of $\vee$ over $\wedge$, is never given as a hint by the system, as it is going down the wrong path. This may lead to a high discriminativity as well, as a student probably only choses to perform this rule, when he thinks he knows it.

6.3 Consulting the domain experts

In this section we present the results of interviewing domain experts, teachers in logic at universities. We ask them to put the rules in the order of difficulty and in the order of discriminativity. We explain the concept of discriminativity on an intuitive level without formulae. These results can be compared to the results we obtain by automated parameter estimation.

We interviewed three domain experts, Daniël Telgen and Leo van Moergestel, who are both senior lecturers at the Utrecht University of Applied Sciences (HU), and prof. Johan Jeuring, supervisor of this thesis, who is one of the main designers and developers of the `ideas` environment, and a professor in computer science at the Dutch Open University.

We first asked in which order of difficulty the rules should be arranged, to their expert opinion. We will describe the order and repeat the arguments the domain experts gave us.

Daniël Telgen thought rules 13 and 14 ($\neg T \implies F$ and $\neg F \implies T$) would be the easiest, as it is the definition of truth, and as such the most basic concept in logic. To his opinion this would be followed by the commutativity laws (rules 1 and 2), as they require an understanding of the and and or operators. The easiest rule after that, he decided ought to be 15, double negation, as it is something that is already contained in elementary school language curricula. The true/false rules would then follow according to mr. Telgen, in this order: 9 and 11 ("according to my intuition "or" is more difficult than "and"."), 10 and 12, 5 and 6 ("also only one thinking step") and then 7 and 8 because the student would have to stop and think a bit whether he/she is dealing with an "or" or an "and". Rules 22 and 23 ought to come after that, because this concept is not paid much attention to in the first courses of the HU curriculum, and is a slightly more difficult concept. Then DeMorgans laws (16 and 17) are followed by distribution (3 and 4). Rule 24 then ought to come after that, as realising a formula is a disjunction takes some thinking, and is a relatively new concept to HU students. According to mr. Telgen he mentions the word disjunction once in the introductory course on computer systems. The most difficult rules he descided ought to be absorption (20 and 21) followed by 18 and then 19. He explicitly mentioned that he found the last four rules very hard to order, but that equivalence elimination, to his intuition seemed more complex than implication elimination. The last four rules all "require quite a bit of thinking to understand well".

The order Daniël Telgen thought would be the difficulty order of the rules would be from low to high: (13,14), (1,2), 15, (9,11), (10,12), (5,6), (7,8), (22,23), (16,17), (3,4), 24, (20,21), 18, 19. The numbers of the rules he thought were of roughly equal difficulty are grouped by brackets. This order is based upon his ideas about how students would understand such rules, and on his knowledge of the HU curriculum. He said that he was most unsure about the order of the last four rules. Discriminativity he found a very difficult concept to judge, so he only made a ranking for difficulty.

The other domain expert from HU we asked to make a ranking for difficulty and discriminativity was Leo van Moergestel. He made the following ranking for difficulty (from low to high): 13, 14, 15, 10, 11, 12, 9, 5, 6, 8, 7, 1, 2, 24, 19, 18, 16, 17, 22, 23, 20, 21, 3, 4. It is interesting to see, that even though Leo van Moergestel and Daniël Telgen both teach at the HU, and both teach computer systems as well as programming they rank

the difficulties of the rules differently. The rules they both rank as difficult are: 3, 4, 16, 17, 18, 19, 20, 21, 22 and 23. Rules 13, 14 and 15 they both consider easy.

Leo van Moergestel also ranked the discriminativity of the rules. From low to high his ranking was: 21, 20, 4, 3, 17, 16, 18, 19, 24, 23, 22, (7, 8, 9, 10, 11, 12), (1, 2, 5, 6), 15, 14, 13.

Johan Jeuring ranked the difficulty of the rules as follows: (13,14), (1,2), (5,6,15), (9,10,11,12), (7,8), (18,23), 22, 19, (3,4,16,17), (20,21). He excluded the ready rule (rule 24) from his ranking.

Both the domain experts and the students who participated in the experiments found discriminativity difficult to judge. We observe that both students and the domain expert who ranked the rules for discriminativity seem to assign low discriminativity to rules they considered difficult, and high discriminativity to rules they considered easy. Discriminativity is a concept which is hard to judge beforehand.

When we compare the difficulty rankings, we see that the domain experts did not come to the exact same rankings. And overview of the rankings is shown in Figure 32.

rule	Name	rewrite rule	diff (1)	diff (2)	diff (3)	disc (2)
1.	Commutativity – and	pΛq ⇒ qΛp	3/4	12	3/4	21
2.	Commutativity – or	pVq ⇒ qVp	3/4	13	3/4	20
3.	Distribution – and over or	pΛ(qVr) ⇒ (pΛq)V(pΛr)	18/19	23	18/19/20/21	4
4.	Distribution – or over and	pV(qΛr) ⇒ (pVq)Λ(pVr)	18/19	24	18/19/20/21	3
5.	Idempotency – and	pΛp ⇒ p	10/11	8	5/6/7	17
6.	Idempotency – or	pVp ⇒ p	10/11	9	5/6/7	16
7.	T/F rules – tautology or	pV¬p ⇒ p	12/13	11	12/13	18
8.	T/F rules – contradiction and	pΛ¬p ⇒ p	12/13	10	12/13	19
9.	T/F rules – and True	pΛT ⇒ p	6/7	7	8/9/10/11	24
10.	T/F rules – or True	pVT ⇒ T	8/9	4	8/9/10/11	23
11.	T/F rules – and False	pΛF ⇒ F	6/7	5	8/9/10/11	22
12.	T/F rules – or False	pVF ⇒ p	8/9	6	8/9/10/11	7
13.	T/F rules – neg True	¬T ⇒ F	1/2	1	1/2	8
14.	T/F rules – neg False	¬F ⇒ T	1/2	2	1/2	9
15.	Double negation	¬¬p ⇒ p	5	3	5/6/7	10
16.	DeMorgan – on Or	¬(pVq) ⇒ ¬pΛ¬q	16/17	17	18/19/20/21	11
17.	DeMorgan – on And	¬(pΛq) ⇒ ¬pV¬q	16/17	18	18/19/20/21	12
18.	Elimination – implication	p→q ⇒ ¬pVq	23	16	14/15	1
19.	Elimination – equivalence	p↔q ⇒ (pΛq)V(¬pΛ¬q)	24	15	17	2
20.	Absorption – outer or	(pΛq)Vp ⇒ p	21/22	21	22/23	5
21.	Absorption – outer and	(pVq)Λp ⇒ p	21/22	22	22/23	6
22.	Tautology F impl	F→p ⇒ T	14/15	19	16	15
23.	Tautology A impl A	p→p ⇒ T	14/15	20	14/15	14
24.	Ready?	pVqVr = ready? ⇒ *yes*	20	14	?	13

Figure 32: The ranking of the rules according to three domain experts: (1) Daniël Telgen (HU), (2) Leo van Moergestel (HU), (3) Johan Jeuring (OU/UU). The first and third experts only ranked the difficulty (diff.) of the rules. The second expert ranked both the difficulty and the discriminativity (disc) of the rules. The rankings are from low to high, starting at 1. Equal rankings are represented by a forward slash.

7 Automated learning from data

In this chapter we apply automated parameter estimation by iterative discrete loglikelihood optimization (IDLLO) to the data. Subsequently we will use the proposed extended IRT model to describe the data, and use cross validation for the model. We compare the outcomes of automatic parameter estimation with the rankings provided by the domain experts. We have shown in the simulations that our proposed model is a reliable model, in the sense that parameter recovery becomes more and more accurate when more data is input. We have also shown that IDLLO can be applied to relatively small amounts of data, with reasonable results. The purpose of this chapter is to evaluate the model we propose against the real-life data we obtained from the `ideas` environment using the `GenExas` interface for the logic domain, and see whether the model is useful to describe and predict real data.

7.1 Using extended IRT as a descriptive model

In this section the data from all students[35] is used to learn the parameters from data. The results are compared to the domain experts' (and our own) expectations. In doing so we hope to show that the proposed model is a suitable descriptor for the data.

We use the model as proposed in equation 14 (section 4.16.2). We run IDLLO using 0 as the minimum value for a and 1.99 as the maximum value, for b the minimum was -5 and the maximum 5, for θ_0 the minimum value was -4 and the maximum value 4 and for η the minimum value was 0 and the maximum value 1. For all parameters the step size is 0.01. We run IDLLO three times, once including the data for all rules, once excluding rules 1, 2 and 14 and once also excluding rule 24 ("ready?").

The item parameters thus estimated are presented in Table 8. As hypothesized the absorption rules are among the top most difficult rules. Rule number 10, $p \vee T \implies T$ however is among the top 3 as well. The easiest rules are also as hypothesized.

In Figure 33 we show the ranking for difficulty resulting from automated parameter estimation (the 20 rules run) next to the rankings provided by the domain experts.

In the discriminativity area we observe that rule 24 and 10 have low discriminativities, as we expected. The absorption rules, implication elimination, equivalence elimination and double negation have high discrimina-

[35] Except for the student who did not not have any positive outcome.

rl.	# o	p.pos.	a (ar)	b (ar)	a (21r)	b (21r)	a (20r)	b (20r)
1	40	1.000	1.99	-5.00	_	_	_	_
2	15	1.000	1.99	-5.00	_	_	_	_
3	263	0.650	0.61	1.70	0.61	1.56	0.49	1.13
4	55	0.764	1.99	1.19	1.99	1.05	1.89	0.46
5	101	0.762	0.59	-0.59	0.59	-0.73	0.59	-1.24
6	91	0.879	0.79	-1.63	0.79	-1.77	0.79	-2.35
7	63	0.635	0.65	0.44	0.65	0.30	0.63	-0.32
8	67	0.776	0.74	-0.33	0.74	-0.47	0.73	-1.08
9	72	0.736	1.06	0.03	1.06	-0.11	1.05	-0.71
10	33	0.424	0.25	2.57	0.25	2.43	0.32	1.54
11	41	0.878	0.98	-1.00	0.98	-1.14	0.98	-1.77
12	115	0.870	1.23	-0.30	1.23	-0.44	1.20	-1.03
13	46	0.913	0.62	-3.14	0.62	-3.28	0.66	-3.70
14	10	1.000	1.99	-5.00	_	_	_	_
15	164	0.902	1.42	-0.97	1.42	-1.11	1.42	-1.65
16	151	0.689	0.85	0.83	0.85	0.69	0.85	0.22
17	116	0.664	1.01	0.96	1.01	0.82	0.90	0.24
18	199	0.714	1.37	0.92	1.37	0.78	1.24	0.28
19	94	0.606	1.05	0.98	1.05	0.84	1.01	0.29
20	56	0.4295	0.63	1.99	0.63	1.85	0.68	1.25
21	55	0.364	0.95	2.29	0.95	2.15	1.02	1.58
22	6	0.333	1.99	0.94	1.99	0.80	1.99	0.14
23	5	0.400	0.99	0.91	0.99	0.77	0.93	0.12
24	290	0.566	0.26	1.55	0.26	1.41	_	_

Table 8: The item parameters estimated using all rules, 21 rules (excluding 1, 2 and 14) and 20 rules (excluding 1,2, 14 and 24). The number of outcomes for the rule is in the column $\#o$, the rule number is $rl.$ and $p.pos.$ is the proportion of positive outcomes.

rule	Name	rewrite rule	diff (1)	diff (2)	diff (3)	diff (ML)
1.	Commutativity – and	p∧q ⇒ q∧p	3/4	12	3/4	1/2/3
2.	Commutativity – or	p∨q ⇒ q∨p	3/4	13	3/4	1/2/3
3.	Distribution – and over or	p∧(q∨r) ⇒ (p∧q)∨(p∧r)	18/19	23	18/19/20/21	20
4.	Distribution – or over and	p∨(q∧r) ⇒ (p∨q)∧(p∨r)	18/19	24	18/19/20/21	19
5.	Idempotency – and	p∧p ⇒ p	10/11	8	5/6/7	8
6.	Idempotency – or	p∨p ⇒ p	10/11	9	5/6/7	5
7.	T/F rules – tautology or	p∨¬p ⇒ p	12/13	11	12/13	12
8.	T/F rules – contradiction and	p∧¬p ⇒ p	12/13	10	12/13	9
9.	T/F rules – and True	p∧T ⇒ p	6/7	7	8/9/10/11	11
10.	T/F rules – or True	p∨T ⇒ T	8/9	4	8/9/10/11	22
11.	T/F rules – and False	p∧F ⇒ F	6/7	5	8/9/10/11	6
12.	T/F rules – or False	p∨F ⇒ p	8/9	6	8/9/10/11	10
13.	T/F rules – neg True	¬T ⇒ F	1/2	1	1/2	4
14.	T/F rules – neg False	¬F ⇒ T	1/2	2	1/2	1/2/3
15.	Double negation	¬¬p ⇒ p	5	3	5/6/7	7
16.	DeMorgan – on Or	¬(p∨q) ⇒ ¬p∧¬q	16/17	17	18/19/20/21	15
17.	DeMorgan – on And	¬(p∧q) ⇒ ¬p∨¬q	16/17	18	18/19/20/21	16
18.	Elimination – implication	p→q ⇒ ¬p∨q	23	16	14/15	17
19.	Elimination – equivalence	p↔q ⇒ (p∧q)∨(¬p∧¬q)	24	15	17	18
20.	Absorption – outer or	(p∧q)∨p ⇒ p	21/22	21	22/23	21
21.	Absorption – outer and	(p∨q)∧p ⇒ p	21/22	22	22/23	23
22.	Tautology F impl	F→p ⇒ T	14/15	19	16	14
23.	Tautology A impl A	p→p ⇒ T	14/15	20	14/15	13
24.	Ready?	p∨q∨r = ready? ⇒ *yes*	20	14	?	?

Figure 33: The difficulty ranking of the rules according to three domain experts: (1) Daniël Telgen (HU), (2) Leo van Moergestel (HU), (3) Johan Jeuring (OU/UU), and the parameters learned by maximum likelihood estimation (ML). The ML ranking is based on the 20 rules run (colum "b (20r)" in Table 8). The three rules which were excluded because they had only positive outcomes, we consider the easiest and are therefore given the lowest rank. The rankings are from low to high, starting at 1. Equal rankings are represented by a forward slash.

tivities. When we compare the rankings of the rules in terms of discriminativity to those provided by the students and domain experts in interviews, we observe that these do not match. There are two reasons for this: (1) discriminativity is a much more difficult concept than difficulty. Discriminativity cannot be percieved on basis of a single observation. While difficulty can be roughly estimated by looking at the percentage of correct applications, discriminativity has no such straightforward heuristic. It needs systematic comparison. Both students and domain experts made remarks in this direction. (2) we know from simulations that the amount of data was too low to estimate discriminativity properly. The parameter estimations for discriminativity have a high uncertainty. We recall Figure 28, where we saw that 8 instances per rule per student yielded very poor results for discriminativity recovery in simulations with 15 students and 23 rules. In the data we had 14 students, 24 rules (of which 4 rules were later discarded), and 6.4 instances per rule per student on average (when we include all rules). (This is 89.5 instances per rule on average for all students.) In order to get more reliable results for discriminativity we need more data.

Comparing the three different runs, it is clear that the third run is closer to the hypotheses. The difficulty of similar rules (e.g. the absorption rules and the reduction with false rules) are closer together.

The learned student parameters θ_0 and η are shown in Table 9. The hypothesis that the UU students have the highest starting competence seems to be confirmed. The UU students have starting competence ranks 1, 3, 4, and 8. The technical computer science students have ranks 2, 6, 9, 10 and 15, and the business informatics students have ranks 5, 7, 11, 12 and 14 in the third run. The order is slightly different in the first two runs.

The learning rate seems to be higher for students with a lower starting competence. We cannot say a lot about the learning speeds however, because we know from simulations that the η estimates have a high uncertainty. We recall Figure 29, where we saw that 8 instances per rule per student yielded very poor results for learning speed recovery in simulations with 15 students and 23 rules. In order to say something about learning speed, when learning this parameter from data we ought to have around 16 instances per rule per student for 15 students and 23 rules. In the experimental data we have only 6.4 instances per rule per student.

From the simulations using IRT we know that difficulty estimates are unreliable at the edges of the student competence range and vice versa. This seems to be the case only for rule 13 and student 14. Note that the competence grows with each application (correct or incorrect) of a rule. The estimates also become more unreliable when there is less data available. The

st.	# o	p.pos.	θ_0 (ar)	η (ar)	θ_0 (21r)	η (21r)	θ_0 (20r)	η (20r)
1	162	0.599	-1.18	0.16	-0.05	0.17	-0.81	0.24
2	157	0.662	-0.99	0.20	0.12	0.22	-0.50	0.24
3	211	0.763	-0.01	0.12	1.18	0.13	0.50	0.17
4	196	0.862	1.48	0.10	2.76	0.11	1.67	0.32
5	140	0.321	-2.18	0.02	-1.15	0.02	-1.79	0.04
6	178	0.927	1.38	0.18	2.69	0.19	2.17	0.16
7	165	0.848	0.70	0.13	1.90	0.15	1.34	0.15
8	181	0.823	0.73	0.07	1.95	0.08	1.38	0.09
9	176	0.739	-0.26	0.21	0.94	0.22	0.19	0.32
10	30	0.500	-1.68	0.49	-0.60	0.52	-1.28	0.58
11	199	0.709	0.18	0.05	1.42	0.05	0.79	0.09
12	149	0.443	-1.58	0.10	-0.51	0.11	-1.20	0.14
13	153	0.765	0.04	0.16	1.25	0.17	0.40	0.28
14	52	0.269	-3.59	0.41	-2.70	0.45	-3.21	0.42

Table 9: The student parameters estimated using all rules, 21 rules (excluding 1, 2 and 14) and 20 rules (excluding 1,2, 14 and 24). The number of outcomes for the rule is in the column $\#o$, the rule number is $rl.$ and $p.pos.$ is the proportion of positive outcomes.

estimates for rules 22 and 23, and student 10 have a higher uncertainty.

Note that with both rules and students, the parameters of difficulty and starting competence do not correspond one to one to the proportions of positive outcomes. The model seems to fit both our and the domain expert's domain knowledge and knowledge about the students, and thus we can say that the model seems to be useful as a descriptive model for e-tutoring in the logic domain, even though the amount of data we had at our disposal to learn the parameters was small.

7.2 Using extended IRT as a predictive model on real data

In this section we will predict the outcomes of the students. In order to do this as in the environment, and thus for practical use, we estimate the parameters as the steps come in. We thus get the data in the form of tuples, in sequence. The first $n-1$ tuples are used to predict the outcome for tuple n (of which the student number and rule number are known).

In order to test the performance of extended IRT on our data set obtained from the logic experiments, we use cross validation. We use a test group, and a training group. Using the training group we obtain the item parameters from simultaneous item and student parameter optimization. For the test group we estimate the student parameters after the first $n-1$ tuples, in order to predict. Because we have very little data we use a test group of size 1 (student). We exclude student 10 and 14 from the test group, as they have only 30 and 52 outcomes. We do however use these students for training. The results of our 12-fold cross validation can be found in Table 10.

The first thing we observe is that the model does not always perform better than just predicting the majority class of the outcomes (usually 1). One reason for this is the skewed distribution of the data - there are more positive outcomes than negative outcomes. This is a well-known obstacle in machine learning. For the one student with a more equal distribution of outcomes, student 12, the model performs significantly better.

Also notice that there are no zero learning speeds. We tested also whether the outcomes would change if negative learning speeds would be allowed. The results were identical. We can therefore conclude that the students do learn something from working with the e-tutoring environment, as their competence rises with the situations they encounter.

The accuracy we get if we take the average of the column containing the accuracy after 50 instances is 0.78, and 0.78 also after 25 instances. The average of the total number of outcomes per rule was 143.5. The simulation results were: an average accuracy of 0.79 for 100 instances and 0.84 for

st.	p.pos.	tot. out.	acc	pabm	p.pos. (a.25)	acc (a.25)	pabm (a.25)	p.pos. (a.50)	acc (a.50)	pabm (a.50)
1	0.60	131	0.63	0.72	0.62	0.63	0.70	0.68	0.60	0.70
2	0.64	125	0.70	0.74	0.64	0.71	0.74	0.67	0.73	0.75
3	0.77	183	0.80	0.83	0.77	0.78	0.83	0.78	0.80	0.83
4	0.91	163	0.89	0.93	0.92	0.92	0.96	0.93	0.93	0.96
5	0.32	114	0.69	0.75	0.31	0.71	0.72	0.34	0.67	0.71
6	0.92	146	0.90	0.91	0.93	0.91	0.91	0.93	0.92	0.91
7	0.85	138	0.83	0.91	0.85	0.85	0.92	0.83	0.82	0.92
8	0.83	157	0.77	0.82	0.83	0.77	0.81	0.82	0.76	0.79
9	0.78	150	0.81	0.89	0.81	0.84	0.89	0.81	0.84	0.90
11	0.72	166	0.69	0.75	0.73	0.69	0.74	0.76	0.67	0.74
12	0.46	125	0.69	0.71	0.46	0.69	0.68	0.52	0.71	0.68
13	0.79	124	0.82	0.86	0.82	0.85	0.86	0.82	0.88	0.85

Table 10: The results of cross validation of using extended IRT as a predictor. st. is the student ID, p.pos. is the total number of positive outcomes of a student divided by the total number of outcomes, tot. out. is the total number of outcomes, acc. is the accuracy and pabm is the predicted accuracy by the model itself (based on the probability of the predictions). The predictions are made based on the item parameters estimated excluding this one student by IDLLO, and the student parameters estimated again after each answer by a student (in sequence). The student parameters are thus updated after each outcome. All values are also shown after the first 25, and the first 50 outcomes - so that the model has received some data with which to estimate the student parameters first.

250 instances. Note that in the simulations we added 1 outcome per rule before we started prediction as well using 24 rules so the total number of outcomes was in fact 124 and 274. The accuracies for the experiments and the simulations are comparable. The accuracy is slightly higher for the simulations, but not significantly.

In Table 10, we see that the model sometimes performs worse, and sometimes better than the accuracy predicted by the model. (The accuracy per student is in the colomns "acc." and the predicted accuracy by the model is in the columns with "pabm" in the title.) This predicted accuracy is obtained from calculating the probability of a right prediction, for each outcome, and calculating the average of that. We assume that this is due to random effects. For some students the model performs significantly worse than the predicted accuracy (students 1 and possibly 11). With this small amount of data however, we cannot form a well-argumented hypothesis as to why this is the case.

From our results, we can say that the model performs according to expectations. For some students however the performance is worse than expected, though we cannot determine if this is a random effect, or a flaw in the model. Also, the data is biased, which makes it hard to make a statement about the adequacy of the model. For now we can say that there is no evidence to reject the validity of extended IRT.

8 Conclusion

The purpose of this research is to create a student model for rule based e-tutoring systems. This model describes and predicts student behaviour: whether or not a student applies the rules correctly. The research question we answer is:

> How can we describe student behaviour in a rule based e-tutoring system with a student model, and estimate the probability of a student applying a rule correctly, the next time he/she tries to apply it?

To answer the research question we have completed several steps (see Figure 1).

We collected data for a specific domain to test the model with. The domain we used was the formal logic domain. Test students were asked to rewrite a given logic expression to disjunctive normal form (DNF). We have developed exercises using 23 rewrite rules. We took effort to select exercises which balance the number of occurences of the rules. We used the `ideas` e-tutoring environment to collect the data.

We selected an existing theory to extend: the two parameter logistic ogive model (2PL) from item response theory (IRT). The 2PL model makes the concepts of difficulty, discriminativity and competence operational by defining a probability function. Difficulty b, discriminativity a and competence θ are the parameters of this function. The probability of a correct outcome - in our case a correct application of a rule - is calculated by filling in specific parameter values for a, b and θ.

We tested 2PL IRT by simulations, focussing on the uncertainty in the item and student parameters first. We find that the uncertainty mainly depends on the match between the range of difficulties of items, and the competence of students. If the difficulty of an item is far from the competence range of the student population the uncertainty in both discriminativity and difficulty grows very large. The same is true for the uncertainty in the student competence when it is far from the difficulty range of the items. In all other cases the uncertainty was acceptable.

We show that 2PL is a suitable predictor for data generated by a 2PL model, and is thus a reliable model. Reliability is of course an important requirement. We observe that after a start up time, the model can predict the outcomes with an adequate accuracy, and that as more data comes in, the predictions become more reliable.

We extend the 2PL model by adding a learning parameter. In order to keep a reliable model (as found by simulations) we need to use a linear model for learning: the student gains a constant amount of competence each time he/she tries to apply a rule. The start competence θ_0 and learning speed η are student parameters. The current competence of a student for a rule is: $\theta_0 + d_i\eta$ where d_i is the number of times the student has encountered rule i sofar. We ran simulations and find that the model was reliable in terms of parameter estimation. We also find that this model could be used on relatively small amounts of data.

We test extended IRT on the real data. We obtained an average accuracy of 0.78, in single students cross-validation. The parameter values that are estimated from simultaneous rule and student parameter estimation match our domain knowledge fairly well. The amount of data we obtained from the experiments however, was small. Only 14 students participated.

The parameter estimations for difficulty correspond very well to the input we got from domain experts, and our own domain knowledge. The domain experts, the students who participated in the experiments and we ourselves find it difficult to rank the rules according to discriminativity. The rankings of domain experts and students did not correspond well to the automated parameter estimations. Furthermore we note that the uncertainties in the discriminativity are very high as we only had an average of 6.4 instances per rule per student and only 14 students.

The start competence of the students as estimated from data, corresponds well to what we know about the different student groups we used in the experiments.

The learning parameter estimates are unreliable, due to the small number of instances per rule. From simulations we know that we need around 16 instances per rule per student in order for reliable learning speed estimates, while we only had 6.4 on average.

We have shown that the accuracy of the extended IRT model we created is adequate and comparable to those accuracies we found by simulation. We have shown by simulation that two out of four parameters cannot be estimated reliably with the amount of data we have. The estimates for the two parameters for which the estimates were reliable according to our simulations, we compared to the input we got from domain experts. Their domain knowledge about the difficulty of the rewrite rules corresponds well to the results we obtained from automated parameter estimation. The estimated start competences of the students correspond to our knowledge about the students.

We therefore conclude that extended IRT is a promising model. More research is required in order to test whether this model is truly suitable for real-life data, but the results we obtained for the logic domain seem to support this conclusion.

9 Future work

In this project we have been trying to teach rules and strategies to students. We have created a competence model based on the rules, which can be applied even to small data sets. In this chapter we will discuss other interesting research options, for which we did not have either the time or the data to do during this project.

Firstly we have created a model that works, but is rather restrictive. Given more data we might be able to relax some of these restrictions. We might also be able to do that if we let students do a small self-test, before working in the e-tutoring environment.

One of the things we would like to predict is which rule a student would chose to apply, in order to see whether the teaching of the strategy is being succesful. We analyzed the data of this project, but we could not discover a pattern that holds over several students. Possible explanations for this are: (a) a "persistent trial and error strategy" may be present (b) students start to think about the strategy after sufficiently mastering the rules. In our experiments there were not so many students who had a sufficient competence. We might discover a pattern once we have analyzed more of these students, or have made the experiment longer. A suggestion by a student was that it might be better to reflect on the strategy after a student has succesfully completed an exercise. This would mean that a student would press the ready button, and then if the result of that action is positive, gets his/her derivation and the suggestions that could be made to improve on the student's strategy.

When working with competence models, a sufficient amount of data is very important. We did our experiments on the basis of voluntary participation, and assumed that many students, who needed the disjunctive normal forms for their exam would join. We were wrong. In fact, of those students only 4 joined the experiments. When doing similar experiments, we would advice to secure the amount of data you will get beforehand, by making the experiments part of the curriculum. We would like to repeat the experiments with a larger amount of students in order to confirm the usefulness of the model, and to be able to test slightly more complex models too.

What we did not do during this project is integrating the competence model into the e-tutoring environment. We would like to see how students react to displaying their estimated competence in different ways. Also, we would like to create a teaching strategy (selecting the right exercise) based on the competence model.

10 Acknowledgments

All voluntary subjects, who are all students at Utrecht University of Applied Sciences, and Utrecht University: Felix Mann, Tim Fennis, Zep Mouris, Wouter Langerak, Dirk Idsingh, Bart van den Boomgaard, Kevin Bakkenes, Guido Passage, Remco Koeijer, Michel Klijn, Simon Liu, Caspar van Tergouw, Nick Rijnbergen and Robbin Stigter, have my deepest gratitude for giving me $2.5 - 3.25$ hours of their time to participate in the experiments. Special thanks to Marleen Schilt, who did not only participate, but also helped to organize the first session.

Prof. Johan Jeuring and dr. Ad Feelders of Utrecht University I thank for their guidance, patience, help and ideas during this project. I would like to thank prof. Jan van Leeuwen, lecturer of the 2010-2011 logic course at Utrecht University as well, for letting me announce the experiment during his lectures, and repeating my call for participants both during the lectures and on the course website.

The domain experts who took a lot of time to help us have my deepest gratitude. Your work is vital to verify the models.

I would also like to thank Emi Yamamoto MA, for taking time to explain educational theories to me, her help and her support, my fellow master students of the software technology laboratory at Utrecht University for their help and good atmosphere at the lab, drs. Christian Bokhove of the Freudenthal institute (Utecht University) whom I discussed a lot with about experimental methods in education, my colleagues and friends at Utrecht University and Utrecht University of Applied Sciences, and my employer (Utrecht University of Applied Sciences) for giving me the opportunity to study alongside my job as a teacher.

11 References

References

[Bak85] Frank B. Baker. *The Basics of Item Response Theory.* Heinemann, 1985.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer Science + Business Media, LLC, 2006.

[BK04] Frank B. Baker and Seock-Ho Kim. *Item Response Theory - Parameter Estimation techniques (2nd ed.).* Taylor and Francis group LLC, Boca Raton, FL, USA, 2004.

[GHJ+07] Alex Gerdes, Bastiaan Heeren, Johan Jeuring, Josje Lodder, Jose Pedro Magalhaes, Arthur van Leeuwen, Harrie Passier, and Sylvia Stuurman. The ideas interactive learning environment, 2007. `http://ideas.cs.uu.nl/trac/`.

[GHJ08] Alex Gerdes, Bastiaan Heeren, and Johan Jeuring. Constructing strategies for programming. Number UU-CS-2008-049, 2008.

[GJH10] Alex Gerdes, Johan Jeuring, and Bastiaan Heeren. Using strategies for assessment of programming exercises. In *Proceedings of the 41st ACM technical symposium on Computer science education, SIGCSE 2010*, pages 441–445, 2010.

[HJ11] Bastiaan Heeren and Johan Jeuring. Interleaving strategies. In *Proceedings of Calculemus/MKM, LNAI 6824*, pages 196–211. Springer, 2011.

[HJG10] Bastiaan Heeren, Johan Jeuring, and Alex Gerdes. Specifying rewrite strategies for interactive exercises. *Mathematics in Computer Science*, 3(3):349–370, 2010.

[HLD82] C. L. Hulin, R. I. Lissak, and F. Drasgow. Recovery of two- and three-parameter logistic item characteristic curves: A monte carlo study. *Applied Psychological Measurement*, 6(3):249–260, 1982.

[JH09] Johan Jeuring and Bastiaan Heeren. An interactive exercise player for math-bridge, 2009.

[Liu09] Xiufeng Liu. *Linking Competence to Opportunities to Learn: Models of Competence and Data Mining*. Springer Publishing Company, Incorporated, 2009.

[Lor80] Frederic M. Lord. *Applications of Item Response Theory to Practical Testing Problems*. Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1980.

[SS10] Leigh Ann Sudol and Cassandra Studer. Analyzing test items: using item response theory to validate assessments. In *SIGCSE '10: Proceedings of the 41st ACM technical symposium on Computer science education*, pages 436–440, New York, NY, USA, 2010. ACM.

[Vyg78] L.S. Vygotsky. *Mind in Society: Development of Higher Psychological Processes*. Harvard University Press, 1978.

A The GenExas interface

In this appendix a screenshot of the GenExas front for the `ideas` environment is presented (figure 34) in order to give an impression of the experimental set-up.

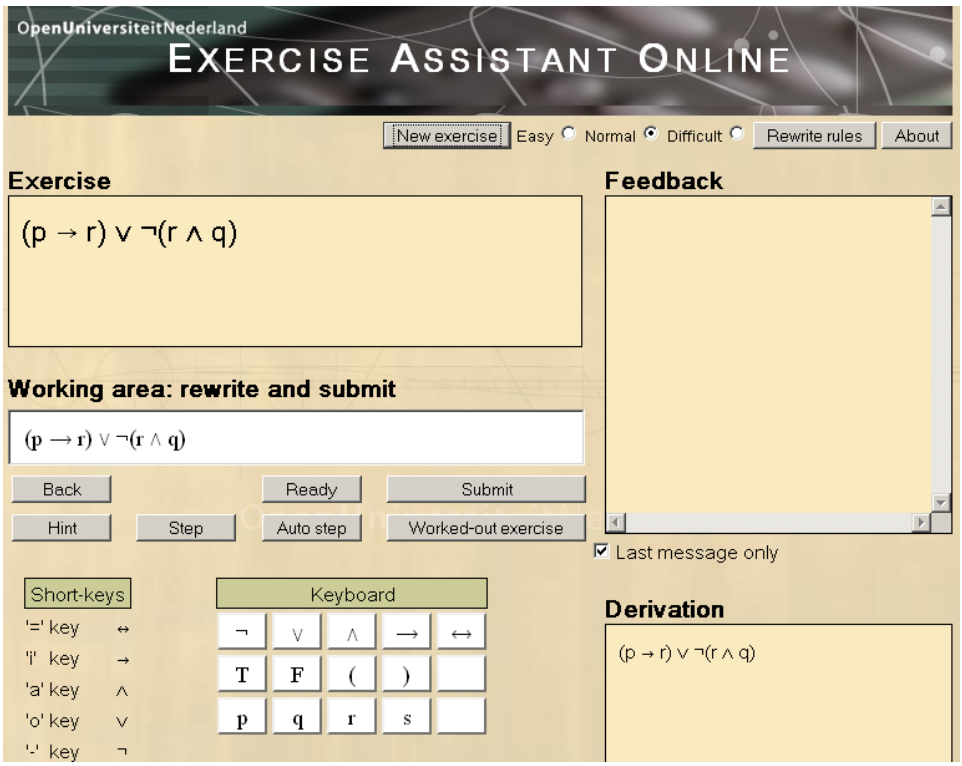


Figure 34: The interface of the DNF e-learning environment used in this experiment.

B Data in tables per student (V0, V1)

In this appendix the data of all students is provided in the form of tables per student (V1). In the caption the tuples[36] (V0) are provided as well. If the data is extraordinarily short, we give a short explanation for why this is. Students 1 to 5 were technical computer science students at Utrecht University of Applied Sciences (Hogeschool Utrecht). Students 6 to 8 were students majoring in computer science at Utrecht University (Universiteit Utrecht). Student number 9 who was doing a minor in computer science at Utrecht University (Universiteit Utrecht). Students 10 to 14 were business informatics students at Utrecht University of Applied Sciences (Hogeschool Utrecht). There was one student who did not get any answer right at the first attempt - as this data cannot be analysed by (extended) IRT, this data is omitted.[37]

[36] See the section on data acquition for explanation about these tuples.

[37] The (start) competence would be minus infinity and the learning parameter cannot be estimated.

1	1 1 1 1 1
2	1 1 1
3	0 0 0 0 0 1 0 1 1 1 0 1 1 1 1 0 1 0
4	0 0 0 0
5	0 1 0 1 1 0 1 1
6	0 1 1 1 1
7	1 0 0 0
8	1 1 0 1
9	0 0 0 1 1 1
10	0 1
11	1 1 1 1 1
12	1 1 1 0 1 1 1 1
13	1 1 1 1
14	
15	0 1 1 1 1 1 1 1 1 1 1
16	1 1 1 0 1 1 1 1 1 1 1 1
17	0 0 0 0 0 1 1 1
18	0 0 0 0 1 0 1 0 0 0 0 1 1 1
19	0 0 0 1 0 1 1 0
20	0 1 1 0 1 0
21	0 1
22	0
23	1
24	1 0 1 1 1 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0
25	

Table 11: data: (1,17,0), (1,15,0), (1,3,0), (1,5,0), (1,24,1), (1,5,1), (1,18,0), (1,13,1), (1,11,1), (1,18,0), (1,15,1), (1,3,0), (1,12,1), (1,24,0), (1,8,1), (1,11,1), (1,15,1), (1,12,1), (1,18,0), (1,24,1), (1,7,1), (1,6,0), (1,19,0), (1,10,0), (1,24,1), (1,13,1), (1,12,1), (1,22,0), (1,24,1), (1,16,1), (1,6,1), (1,18,0), (1,17,0), (1,15,1), (1,15,1), (1,6,1), (1,2,1), (1,24,1), (1,18,1), (1,19,0), (1,16,1), (1,15,1), (1,8,1), (1,11,1), (1,24,0), (1,7,0), (1,12,0), (1,3,0), (1,15,1), (1,12,1), (1,17,0), (1,21,0), (1,24,0), (1,3,0), (1,5,0), (1,9,0), (1,17,0), (1,18,0), (1,8,0), (1,11,1), (1,3,0), (1,23,1), (1,6,1), (1,13,1), (1,24,0), (1,15,1), (1,19,0), (1,3,1), (1,20,0), (1,5,1), (1,19,1), (1,24,0), (1,18,1), (1,7,0), (1,10,1), (1,9,0), (1,3,0), (1,9,0), (1,9,1), (1,24,1), (1,21,1), (1,17,0), (1,7,0), (1,9,1), (1,16,1), (1,4,0), (1,24,0), (1,18,0), (1,19,0), (1,17,1), (1,16,0), (1,15,1), (1,24,1), (1,18,0), (1,6,1), (1,13,1), (1,11,1), (1,12,1), (1,24,0), (1,16,1), (1,9,1), (1,12,1), (1,3,1), (1,24,1), (1,19,1), (1,16,1), (1,1,1), (1,17,1), (1,1,1), (1,5,1), (1,1,1), (1,3,1), (1,5,0), (1,24,0), (1,4,0), (1,20,1), (1,20,1), (1,20,0), (1,3,1), (1,3,0), (1,5,1), (1,18,0), (1,16,1), (1,24,0), (1,3,1), (1,18,0), (1,3,1), (1,8,1), (1,12,1), (1,24,0), (1,2,1), (1,18,1), (1,19,1), (1,3,1), (1,24,1), (1,18,1), (1,16,1), (1,15,1), (1,24,0), (1,3,1), (1,18,1), (1,16,1), (1,16,1), (1,4,0), (1,20,1), (1,5,1), (1,24,1), (1,19,0), (1,16,1), (1,17,1), (1,3,0), (1,24,0), (1,20,0), (1,4,0), (1,1,1), (1,3,1), (1,16,1), (1,15,1), (1,2,1), (1,24,0), (1,3,0), (1,1,1)

1	1 1 1 1 1
2	1 1
3	0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1
4	0 1
5	1 1 1 1 0
6	1 1 1 1 1 1 1
7	1 0 1 0
8	0 1 0 0 1 1 1
9	1 1 0 1
10	0 0 0
11	1 0 1
12	1 1 1 0 1 1 1 1 1
13	1 1 1
14	1
15	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
16	1 0 0 0 1 0 1 0 1
17	1 1 0 1 0
18	0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1
19	0 1 0 1 1 0 0 1 1
20	0
21	0 0 0
22	0
23	
24	1 0 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
25	

Table 12: data: (2,17,1), (2,15,1), (2,15,1), (2,18,0), (2,24,1), (2,18,0), (2,15,1), (2,15,1), (2,21,0), (2,24,0), (2,3,0), (2,6,1), (2,5,1), (2,18,0), (2,15,1), (2,3,1), (2,24,0), (2,13,1), (2,8,0), (2,15,1), (2,18,0), (2,11,1), (2,24,1), (2,7,1), (2,9,1), (2,6,1), (2,19,0), (2,24,0), (2,22,0), (2,18,1), (2,14,1), (2,12,1), (2,10,0), (2,24,1), (2,18,0), (2,6,1), (2,15,1), (2,6,1), (2,24,1), (2,18,1), (2,19,1), (2,24,0), (2,16,1), (2,15,1), (2,8,1), (2,11,0), (2,12,1), (2,8,0), (2,3,0), (2,8,0), (2,12,1), (2,15,1), (2,12,0), (2,17,1), (2,21,0), (2,24,1), (2,9,1), (2,16,0), (2,7,0), (2,10,0), (2,18,1), (2,21,0), (2,1,1), (2,8,1), (2,24,1), (2,6,1), (2,18,1), (2,7,1), (2,13,1), (2,12,1), (2,24,1), (2,15,1), (2,19,0), (2,20,0), (2,19,1), (2,24,1), (2,18,1), (2,7,0), (2,10,0), (2,9,0), (2,24,1), (2,19,1), (2,18,1), (2,17,0), (2,16,0), (2,15,1), (2,24,1), (2,13,1), (2,11,1), (2,12,1), (2,18,1), (2,6,1), (2,18,1), (2,24,1), (2,12,1), (2,9,1), (2,3,0), (2,1,1), (2,24,1), (2,19,0), (2,24,0), (2,3,0), (2,4,0), (2,19,0), (2,5,1), (2,16,0), (2,17,1), (2,1,1), (2,5,1), (2,3,1), (2,5,1), (2,18,1), (2,16,1), (2,3,0), (2,24,1), (2,18,1), (2,3,0), (2,24,0), (2,8,1), (2,12,1), (2,3,0), (2,18,1), (2,19,1), (2,3,0), (2,24,1), (2,18,1), (2,16,0), (2,15,1), (2,3,0), (2,24,0), (2,18,1), (2,16,1), (2,15,1), (2,6,1), (2,24,1), (2,19,1), (2,24,0), (2,3,0), (2,16,0), (2,17,0), (2,3,0), (2,3,1), (2,16,1), (2,15,1), (2,2,1), (2,4,1), (2,3,1), (2,2,1), (2,1,1), (2,1,1), (2,3,1), (2,5,0), (2,3,1), (2,3,1), (2,8,1), (2,12,1), (2,24,1)

1	1 1
2	
3	0 1 0 0 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1
4	0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
5	1 1 0 1 1 0 0 1 0
6	1 1 1 1 0 1 0 1 1 1
7	1 1 1 1 1 1 0
8	1 1 0 1 0 1 1
9	1 1 1 1 1 1 1 1 1
10	1 1 1
11	1 1 1 1
12	1 1 1 1 1 1 1 1 1 1 1 1 1 1
13	1 1 1 1
14	1
15	1 1 1 1 1 1 1 1 1 1 1 1 1
16	0 0 1 0 1 1 1 1 0 1
17	0 0 1 0 1 1 1 0 0 1 1
18	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
19	0 1 1 1 1 1 1 1
20	0 1 0
21	0 0 0 0 0 0
22	1
23	0
24	0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 0 0 1 0 1 1 0 0
25	

Table 13: data: (3,17,0), (3,15,1), (3,15,1), (3,21,0), (3,24,0), (3,3,0), (3,5,1), (3,18,0), (3,15,1), (3,13,1), (3,11,1), (3,12,1), (3,3,1), (3,24,1), (3,15,1), (3,8,1), (3,11,1), (3,12,1), (3,18,1), (3,24,1), (3,7,1), (3,9,1), (3,6,1), (3,19,0), (3,24,1), (3,13,1), (3,12,1), (3,18,1), (3,14,1), (3,22,1), (3,6,1), (3,18,1), (3,15,1), (3,6,1), (3,1,1), (3,24,1), (3,18,1), (3,19,1), (3,16,0), (3,3,0), (3,8,1), (3,12,1), (3,16,0), (3,15,1), (3,8,0), (3,11,1), (3,12,1), (3,24,1), (3,15,1), (3,12,1), (3,17,0), (3,21,0), (3,24,0), (3,9,1), (3,17,1), (3,24,1), (3,18,1), (3,3,0), (3,24,1), (3,6,1), (3,23,0), (3,18,1), (3,7,1), (3,13,1), (3,12,1), (3,24,1), (3,15,1), (3,4,0), (3,6,0), (3,5,1), (3,1,1), (3,3,1), (3,5,0), (3,4,1), (3,5,1), (3,19,1), (3,17,0), (3,24,0), (3,3,1), (3,5,1), (3,19,1), (3,20,0), (3,20,1), (3,18,1), (3,7,1), (3,10,1), (3,9,1), (3,24,1), (3,17,1), (3,16,1), (3,4,1), (3,3,1), (3,5,0), (3,7,1), (3,10,1), (3,9,1), (3,24,1), (3,3,0), (3,17,1), (3,17,1), (3,21,0), (3,3,0), (3,18,1), (3,3,1), (3,19,1), (3,16,0), (3,17,0), (3,24,1), (3,17,0), (3,15,1), (3,17,1), (3,3,0), (3,21,0), (3,3,1), (3,13,1), (3,11,1), (3,12,1), (3,6,1), (3,18,1), (3,16,1), (3,24,1), (3,12,1), (3,9,1), (3,3,1), (3,24,1), (3,19,1), (3,4,1), (3,4,1), (3,24,1), (3,3,0), (3,18,1), (3,4,0), (3,5,0), (3,16,1), (3,4,1), (3,3,0), (3,3,1), (3,24,0), (3,3,0), (3,24,0), (3,18,1), (3,3,1), (3,3,1), (3,3,1), (3,8,1), (3,12,1), (3,3,1), (3,5,1), (3,24,1), (3,19,1), (3,4,1), (3,4,1), (3,18,1), (3,3,1), (3,4,1), (3,7,1), (3,9,1), (3,7,1), (3,8,0), (3,9,1), (3,12,1), (3,24,0), (3,3,0), (3,21,0), (3,3,1), (3,3,1), (3,8,1), (3,12,1), (3,3,1), (3,8,1), (3,12,1), (3,18,1), (3,16,1), (3,15,1), (3,3,0), (3,3,1), (3,24,1), (3,18,1), (3,6,0), (3,16,1), (3,15,1), (3,24,1), (3,4,1), (3,3,1), (3,3,1), (3,3,1), (3,5,0), (3,19,1), (3,4,1), (3,6,1), (3,3,1), (3,6,1), (3,4,1), (3,3,1), (3,16,0), (3,20,0), (3,4,1), (3,6,1), (3,24,0), (3,16,1), (3,15,1), (3,24,0), (3,4,1), (3,4,1), (3,7,0), (3,17,1), (3,3,1), (3,21,0), (3,3,1), (3,10,1), (3,9,1)

1	1 1 1
2	1 1 1
3	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
4	1 1 1 1 1 1 1
5	0 1 1 1 1 1 1 1 1 1 1 1
6	1 1 1 1 1 1 1 1
7	1 1 0 1 1
8	1 1 1 1 1 1
9	1 1 1 1 1 1
10	0 0 0
11	1 1 1 1
12	1 1 1 1 1 1 1 1 1 1 1
13	1 1 1 1
14	1
15	1 1 1 1 1 1 1 1 1 1 1 1 1
16	0 1 1 1 1 1 1 1 1 1 1
17	1 0 1 1 1 1 1 1 0 1 1
18	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
19	1 1 1 0 1 1 1 1 1
20	0 0 1 0
21	0 0 1 1 1
22	
23	
24	0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 1 1 0 0 0 0
25	

Table 14: data:(4,17,1), (4,15,1), (4,15,1), (4,21,0), (4,24,0), (4,3,1), (4,5,0), (4,20,0), (4,17,0), (4,5,1), (4,13,1), (4,11,1), (4,12,1), (4,24,0), (4,18,1), (4,15,1), (4,3,1), (4,3,1), (4,3,1), (4,15,1), (4,8,1), (4,11,1), (4,12,1), (4,18,1), (4,24,1), (4,7,1), (4,9,1), (4,6,1), (4,19,1), (4,24,1), (4,13,1), (4,12,1), (4,18,1), (4,14,1), (4,10,0), (4,24,1), (4,6,1), (4,16,0), (4,18,1), (4,17,1), (4,15,1), (4,15,1), (4,2,1), (4,24,1), (4,18,1), (4,19,1), (4,21,0), (4,16,1), (4,15,1), (4,8,1), (4,11,1), (4,12,1), (4,3,1), (4,8,1), (4,12,1), (4,24,1), (4,15,1), (4,12,1), (4,17,1), (4,24,0), (4,3,1), (4,2,1), (4,9,1), (4,17,1), (4,17,1), (4,24,1), (4,18,1), (4,3,1), (4,3,1), (4,8,1), (4,24,1), (4,6,1), (4,20,0), (4,18,1), (4,7,1), (4,13,1), (4,12,1), (4,24,1), (4,15,1), (4,3,1), (4,5,1), (4,5,1), (4,1,1), (4,3,1), (4,6,1), (4,3,1), (4,5,1), (4,20,1), (4,19,1), (4,24,1), (4,18,1), (4,7,0), (4,10,0), (4,9,1), (4,24,1), (4,17,1), (4,21,1), (4,7,1), (4,9,1), (4,16,1), (4,24,1), (4,19,0), (4,18,1), (4,17,1), (4,16,1), (4,15,1), (4,19,1), (4,24,1), (4,13,1), (4,11,1), (4,12,1), (4,6,1), (4,18,1), (4,24,0), (4,16,1), (4,9,1), (4,12,1), (4,24,0), (4,3,1), (4,4,1), (4,4,1), (4,19,1), (4,17,0), (4,19,1), (4,16,1), (4,17,1), (4,24,0), (4,1,1), (4,5,1), (4,3,1), (4,2,1), (4,21,1), (4,3,1), (4,5,1), (4,18,1), (4,16,1), (4,24,0), (4,3,1), (4,18,1), (4,3,1), (4,8,1), (4,12,1), (4,24,0), (4,3,1), (4,5,1), (4,3,1), (4,5,1), (4,24,1), (4,5,1), (4,5,1), (4,3,1), (4,3,1), (4,18,1), (4,3,1), (4,8,1), (4,12,1), (4,19,1), (4,24,1), (4,18,1), (4,16,1), (4,15,1), (4,24,0), (4,3,1), (4,3,1), (4,6,1), (4,18,1), (4,16,1), (4,15,1), (4,4,1), (4,4,1), (4,6,1), (4,24,0), (4,21,1), (4,19,1), (4,16,1), (4,17,1), (4,3,1), (4,24,0), (4,20,0), (4,1,1), (4,3,0), (4,16,1), (4,15,1), (4,4,1), (4,4,1), (4,7,1), (4,10,0), (4,9,1), (4,24,0), (4,3,1), (4,3,1), (4,5,1), (4,4,1), (4,6,1), (4,3,1), (4,3,1)

1	1 1 1
2	
3	0 1 0 0 0 0 0 0 1 0 0 0
4	
5	0 0 0 0 1 1 1 1 1
6	1 1 1 1 1 1
7	1 0 1
8	1 0 1
9	0 1 0 0 0 1 0
10	0 0
11	1 0 0
12	0 0 0 0 0 1 0 1
13	1 1 1 1
14	1
15	0 0 1 0 1 1 1 0 1 1 0 0 0 1 1
16	0 0 0 0 0 0 0 0
17	0 0 0 0 0 0 1
18	0 0 0 0 0 0 0 0 0 0 0 0 0 0
19	0 0 1 0 0 0 0
20	0 1 0
21	0 0 0
22	
23	0
24	0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1
25	

Table 15: data:(5,17,0), (5,24,0), (5,3,0), (5,15,0), (5,15,0), (5,5,0), (5,5,0), (5,16,0), (5,5,0), (5,24,0), (5,9,0), (5,13,1), (5,11,1), (5,12,0), (5,18,0), (5,15,1), (5,3,1), (5,15,0), (5,24,0), (5,8,1), (5,11,0), (5,12,0), (5,18,0), (5,6,1), (5,7,1), (5,24,0), (5,9,1), (5,19,0), (5,13,1), (5,12,0), (5,18,0), (5,24,0), (5,14,1), (5,6,1), (5,15,1), (5,24,1), (5,19,0), (5,18,0), (5,24,0), (5,18,0), (5,15,1), (5,12,0), (5,15,1), (5,24,0), (5,17,0), (5,3,0), (5,15,0), (5,21,0), (5,17,0), (5,9,0), (5,17,0), (5,5,0), (5,24,0), (5,9,0), (5,10,0), (5,18,0), (5,24,0), (5,3,0), (5,6,1), (5,23,0), (5,13,1), (5,12,0), (5,15,1), (5,20,0), (5,5,1), (5,15,1), (5,19,1), (5,24,1), (5,18,0), (5,7,0), (5,10,0), (5,9,0), (5,24,1), (5,21,0), (5,7,1), (5,9,1), (5,24,0), (5,17,0), (5,16,0), (5,18,0), (5,24,0), (5,17,0), (5,19,0), (5,6,1), (5,13,1), (5,11,0), (5,12,1), (5,18,0), (5,24,0), (5,16,0), (5,9,0), (5,12,0), (5,24,0), (5,3,0), (5,19,0), (5,3,0), (5,24,0), (5,16,0), (5,5,1), (5,20,1), (5,20,0), (5,16,0), (5,17,1), (5,6,1), (5,3,0), (5,1,1), (5,5,1), (5,1,1), (5,5,1), (5,5,1), (5,3,0), (5,18,0), (5,24,0), (5,18,0), (5,8,0), (5,3,1), (5,8,1), (5,12,1), (5,21,0), (5,19,0), (5,24,0), (5,18,0), (5,18,0), (5,15,0), (5,16,0), (5,15,0), (5,24,0), (5,3,0), (5,1,1), (5,6,1), (5,15,0), (5,16,0), (5,15,1), (5,19,0), (5,3,0), (5,24,0), (5,16,0), (5,15,1), (5,3,0), (5,24,1)

1	1 1 1 1
2	1 1
3	1 0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1
4	1 1
5	1 1 1 1 1 1 1
6	1 1 1 1 1 1 1
7	1 1 1 1 1
8	1 1 1 1 1 1 1
9	1 1 1 1 1 1
10	1 1 1
11	1 1 1
12	1 1 1 1 1 1 1 1 1 1 1 1
13	1 1 1
14	1
15	1 1 1 1 1 1 1 1 1 1 1 1 1
16	1 0 1 1 1 1 1 1 1 1 0
17	1 1 1 1 1 1 1 1 1 1
18	0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
19	0 1 1 1 1 1
20	1 0 1 1 1
21	0 1 1 0
22	
23	
24	1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
25	1

Table 16: data:(6,17,1), (6,15,1), (6,15,1), (6,1,1), (6,3,1), (6,5,1), (6,20,1), (6,24,1), (6,17,1), (6,6,1), (6,18,0), (6,15,1), (6,13,1), (6,11,1), (6,12,1), (6,1,1), (6,3,0), (6,15,1), (6,8,1), (6,11,1), (6,12,1), (6,18,1), (6,24,1), (6,7,1), (6,9,1), (6,6,1), (6,25,1), (6,18,1), (6,18,1), (6,3,0), (6,3,1), (6,3,1), (6,8,1), (6,12,1), (6,8,1), (6,12,1), (6,24,1), (6,18,1), (6,14,1), (6,10,1), (6,24,1), (6,16,1), (6,6,1), (6,18,1), (6,17,1), (6,15,1), (6,15,1), (6,2,1), (6,6,1), (6,24,1), (6,18,1), (6,19,0), (6,16,0), (6,15,1), (6,8,1), (6,11,1), (6,24,1), (6,3,1), (6,8,1), (6,12,1), (6,24,1), (6,15,1), (6,12,1), (6,17,1), (6,3,1), (6,5,1), (6,20,0), (6,24,1), (6,9,1), (6,17,1), (6,7,1), (6,10,1), (6,24,1), (6,18,1), (6,3,1), (6,8,1), (6,12,1), (6,24,1), (6,18,1), (6,7,1), (6,13,1), (6,12,1), (6,6,1), (6,20,1), (6,15,1), (6,20,1), (6,5,1), (6,19,1), (6,24,1), (6,18,1), (6,7,1), (6,10,1), (6,9,1), (6,24,1), (6,3,1), (6,21,0), (6,7,1), (6,9,1), (6,16,1), (6,17,1), (6,17,1), (6,3,1), (6,21,1), (6,3,0), (6,24,1), (6,18,1), (6,17,1), (6,16,1), (6,15,1), (6,19,1), (6,4,1), (6,24,1), (6,24,0), (6,6,1), (6,18,0), (6,16,1), (6,13,1), (6,4,1), (6,12,1), (6,24,1), (6,9,1), (6,12,1), (6,3,1), (6,24,1), (6,19,1), (6,16,1), (6,17,1), (6,2,1), (6,21,1), (6,3,1), (6,3,1), (6,3,0), (6,1,1), (6,5,1), (6,1,1), (6,5,1), (6,24,1), (6,5,1), (6,18,1), (6,16,1), (6,3,1), (6,24,1), (6,18,1), (6,21,0), (6,3,1), (6,9,1), (6,12,1), (6,3,1), (6,5,1), (6,24,1), (6,18,1), (6,3,1), (6,8,1), (6,12,1), (6,19,1), (6,24,1), (6,18,1), (6,16,1), (6,15,1), (6,3,1), (6,24,1), (6,6,1), (6,18,1), (6,16,1), (6,15,1), (6,20,1), (6,19,1), (6,16,1), (6,17,1), (6,3,1), (6,3,1), (6,3,1), (6,24,1), (6,16,0), (6,15,1), (6,3,1), (6,3,1), (6,24,1)

1	1 1
2	1 1
3	1 1 0 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 0 0 1
4	1 1 0 1 1 1 1
5	0 0 1 1 0 1 1 1
6	1 1 1 1 1 1 1
7	1 0 1 1
8	1 1 1 1
9	1 1 1 1 1
10	1
11	1 1 1 1
12	0 1 1 1 1 1 1
13	1 1 1 1
14	
15	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
16	1 1 1 1 0 1 1 1 1 1 1 1 0
17	1 1 1 1 1 1 1 1 1 1 1 1 1
18	0 1 1 1 1 1 1 1 1 1 1 1 1
19	1 1 1 1 0 1 1
20	0 0 0 1
21	0 0
22	1
23	1
24	0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1
25	

Table 17: data:(7,17,1), (7,15,1), (7,15,1), (7,3,1), (7,17,1), (7,18,0), (7,24,0), (7,16,1), (7,15,1), (7,5,0), (7,4,1), (7,13,1), (7,11,1), (7,12,0), (7,5,0), (7,3,1), (7,15,1), (7,8,1), (7,11,1), (7,12,1), (7,18,1), (7,24,1), (7,7,1), (7,6,1), (7,19,1), (7,9,1), (7,24,1), (7,13,1), (7,12,1), (7,22,1), (7,24,1), (7,6,1), (7,16,1), (7,18,1), (7,17,1), (7,15,1), (7,15,1), (7,2,1), (7,6,1), (7,24,1), (7,18,1), (7,19,1), (7,3,0), (7,8,1), (7,12,1), (7,17,1), (7,15,1), (7,8,1), (7,11,1), (7,15,1), (7,12,1), (7,17,1), (7,24,0), (7,21,0), (7,3,1), (7,9,1), (7,17,1), (7,7,0), (7,24,1), (7,18,1), (7,3,1), (7,24,1), (7,6,1), (7,23,1), (7,13,1), (7,24,1), (7,15,1), (7,4,1), (7,6,1), (7,3,1), (7,5,1), (7,20,0), (7,1,1), (7,5,1), (7,20,0), (7,3,1), (7,5,0), (7,19,1), (7,24,1), (7,18,1), (7,7,1), (7,10,1), (7,9,1), (7,24,1), (7,3,1), (7,16,1), (7,17,1), (7,17,1), (7,21,0), (7,7,1), (7,9,1), (7,3,0), (7,24,1), (7,18,1), (7,17,1), (7,16,1), (7,15,1), (7,19,1), (7,24,1), (7,13,1), (7,11,1), (7,6,1), (7,18,1), (7,16,0), (7,24,1), (7,9,1), (7,12,1), (7,3,1), (7,24,1), (7,19,0), (7,16,1), (7,17,1), (7,1,1), (7,5,1), (7,24,0), (7,3,0), (7,16,1), (7,5,1), (7,17,1), (7,20,0), (7,3,1), (7,5,1), (7,18,1), (7,16,1), (7,4,0), (7,3,0), (7,24,1), (7,18,1), (7,2,1), (7,4,1), (7,3,1), (7,24,1), (7,18,1), (7,3,1), (7,8,1), (7,12,1), (7,19,1), (7,24,1), (7,18,1), (7,16,1), (7,15,1), (7,24,0), (7,3,1), (7,3,1), (7,6,1), (7,18,1), (7,16,1), (7,15,1), (7,20,1), (7,24,1), (7,4,1), (7,19,1), (7,4,1), (7,3,1), (7,4,1), (7,17,1), (7,16,1), (7,16,1), (7,3,0), (7,24,1), (7,16,0), (7,15,1), (7,3,0), (7,3,1), (7,24,1)

1	
2	
3	1 1 1 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 0 1
4	1 1 1 0 1
5	1 1 1 1 1
6	1 1 1 1 1 1
7	1 1 1 1 1 1 0 1
8	1 1 0 1 1
9	0 1 1 1 1 0
10	0 0 1 0
11	1 1 1
12	1 1 1 1 1 1 1 1 1 1 1 1
13	1 1 1 1 1
14	1
15	1 1 1 1 1 1 1 0 1 1 1
16	1 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1
17	0 0 1 1 1 1 1 1 0 1
18	1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
19	0 1 1 1 0 1 1
20	1 1 1 1 1
21	1 1 1 0 1 1
22	
23	
24	1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 0 0
25	

Table 18: data:(8,16,1), (8,17,0), (8,15,1), (8,21,1), (8,24,1), (8,5,1), (8,13,1), (8,11,1), (8,12,1), (8,18,1), (8,15,1), (8,3,1), (8,24,1), (8,15,1), (8,8,1), (8,11,1), (8,12,1), (8,18,1), (8,6,1), (8,7,1), (8,9,0), (8,19,0), (8,13,1), (8,12,1), (8,18,1), (8,14,1), (8,10,0), (8,24,1), (8,6,1), (8,18,1), (8,15,1), (8,6,1), (8,24,1), (8,18,1), (8,19,1), (8,16,0), (8,3,1), (8,8,1), (8,12,1), (8,16,0), (8,17,0), (8,7,1), (8,10,0), (8,13,1), (8,12,1), (8,24,1), (8,15,1), (8,17,1), (8,12,1), (8,3,1), (8,21,1), (8,24,1), (8,9,1), (8,17,1), (8,17,1), (8,7,1), (8,10,1), (8,24,1), (8,18,0), (8,3,0), (8,24,0), (8,8,0), (8,12,1), (8,3,1), (8,18,1), (8,7,1), (8,13,1), (8,6,1), (8,12,1), (8,24,1), (8,15,1), (8,19,1), (8,20,1), (8,20,1), (8,5,1), (8,5,1), (8,24,1), (8,18,1), (8,7,1), (8,10,0), (8,9,1), (8,24,1), (8,21,1), (8,7,1), (8,9,1), (8,17,1), (8,24,0), (8,16,0), (8,18,1), (8,19,1), (8,17,1), (8,16,1), (8,15,1), (8,7,0), (8,15,0), (8,24,1), (8,6,1), (8,13,1), (8,11,1), (8,12,1), (8,18,1), (8,16,1), (8,24,1), (8,12,1), (8,9,1), (8,3,1), (8,24,1), (8,4,1), (8,4,1), (8,19,0), (8,21,0), (8,4,1), (8,3,0), (8,5,1), (8,16,1), (8,16,1), (8,16,1), (8,20,1), (8,20,1), (8,16,1), (8,16,1), (8,17,1), (8,24,0), (8,3,0), (8,5,1), (8,18,1), (8,16,1), (8,17,0), (8,3,1), (8,16,1), (8,3,0), (8,16,0), (8,24,1), (8,18,1), (8,3,1), (8,8,1), (8,12,1), (8,3,0), (8,21,1), (8,24,1), (8,19,1), (8,18,1), (8,3,1), (8,8,1), (8,12,1), (8,24,1), (8,18,1), (8,16,0), (8,15,1), (8,3,1), (8,3,1), (8,24,1), (8,18,1), (8,6,1), (8,16,1), (8,15,1), (8,20,1), (8,24,1), (8,19,1), (8,3,1), (8,16,1), (8,4,0), (8,16,1), (8,17,1), (8,16,1), (8,16,1), (8,24,0), (8,3,1), (8,16,1), (8,15,1), (8,3,1), (8,16,1), (8,3,1), (8,16,1), (8,4,1), (8,7,1), (8,9,0), (8,24,0), (8,3,0), (8,21,1), (8,3,1)

1	1 1 1
2	1
3	0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1
4	1 0 1 0 1 1
5	1 1 1 1 1 1 1 1 1 1 1 1 1
6	1 1 1 1 1 1 1 1
7	0 0 0 0
8	1 1 1 1 1 1
9	0 1 1 1 1
10	0 1 0
11	1 1 1
12	0 1 1 1 1 1 1 1 1 1
13	1 0
14	1
15	1 1 1 1 1 1 1 1 1 1 1 1 1 1
16	0 1 1 1 0 1 0 0 1 1 1
17	0 1 0 1 1 0 1 1 1 1
18	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
19	0 0 1 1 0 1 0
20	0 0 0
21	1 0 0 1 1 1 0
22	
23	
24	0 0 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 0 0 1 0
25	

Table 19: data:(9,17,0), (9,15,1), (9,15,1), (9,21,1), (9,24,0), (9,13,1), (9,11,1), (9,12,0), (9,18,1), (9,15,1), (9,5,1), (9,3,0), (9,24,0), (9,18,1), (9,15,1), (9,8,1), (9,11,1), (9,12,1), (9,7,0), (9,24,1), (9,7,0), (9,24,0), (9,6,1), (9,9,0), (9,19,0), (9,18,1), (9,14,1), (9,10,0), (9,24,0), (9,13,0), (9,12,1), (9,6,1), (9,18,1), (9,15,1), (9,2,1), (9,6,1), (9,24,1), (9,18,1), (9,19,0), (9,3,0), (9,10,1), (9,15,1), (9,8,1), (9,11,1), (9,8,1), (9,12,1), (9,12,1), (9,24,1), (9,15,1), (9,24,0), (9,12,1), (9,17,1), (9,21,0), (9,3,1), (9,5,1), (9,9,1), (9,17,0), (9,24,1), (9,18,1), (9,3,0), (9,24,0), (9,6,1), (9,18,1), (9,16,0), (9,15,1), (9,8,1), (9,4,1), (9,24,0), (9,6,1), (9,12,1), (9,3,1), (9,15,1), (9,3,1), (9,5,1), (9,1,1), (9,5,1), (9,20,0), (9,19,1), (9,16,1), (9,17,1), (9,3,1), (9,4,0), (9,5,1), (9,3,1), (9,1,1), (9,5,1), (9,4,1), (9,6,1), (9,24,0), (9,5,1), (9,3,1), (9,5,1), (9,6,1), (9,18,1), (9,7,0), (9,10,0), (9,9,1), (9,17,1), (9,21,0), (9,4,0), (9,3,1), (9,21,1), (9,16,1), (9,7,0), (9,9,1), (9,24,1), (9,18,1), (9,3,1), (9,16,1), (9,17,0), (9,19,1), (9,17,1), (9,15,1), (9,17,1), (9,3,1), (9,21,1), (9,3,1), (9,24,1), (9,9,1), (9,12,1), (9,3,1), (9,24,1), (9,19,0), (9,3,1), (9,5,1), (9,20,0), (9,20,0), (9,16,0), (9,1,1), (9,5,1), (9,17,1), (9,24,0), (9,3,1), (9,5,1), (9,18,1), (9,16,1), (9,4,1), (9,3,0), (9,24,0), (9,21,1), (9,18,1), (9,3,1), (9,8,1), (9,12,1), (9,21,0), (9,24,0), (9,18,1), (9,3,1), (9,8,1), (9,24,0), (9,12,1), (9,19,1), (9,18,1), (9,16,0), (9,15,1), (9,16,0), (9,18,1), (9,16,1), (9,15,1), (9,6,1), (9,24,1), (9,19,0), (9,4,1), (9,3,0), (9,3,1), (9,3,1), (9,3,1), (9,24,0), (9,5,1), (9,16,1), (9,17,1), (9,3,1), (9,3,1), (9,16,1), (9,15,1), (9,3,1)

1	
2	
3	
4	
5	1
6	0 0 1 0 1 1
7	
8	
9	1
10	
11	
12	
13	
14	
15	0 1 1 1
16	0 0
17	0 1 0 0
18	1 1 1 0 1 1 1
19	0 0
20	
21	0
22	0
23	0
24	
25	

Table 20: data:(10,17,0), (10,15,0), (10,17,1), (10,6,0), (10,15,1), (10,6,0), (10,22,0), (10,6,1), (10,16,0), (10,18,1), (10,15,1), (10,17,0), (10,18,1), (10,6,0), (10,23,0), (10,15,1), (10,18,1), (10,21,0), (10,17,0), (10,18,0), (10,6,1), (10,9,1), (10,19,0), (10,5,1), (10,18,1), (10,18,1), (10,18,1), (10,6,1), (10,19,0), (10,16,0)
This student misinterpreted the experimental instructions, and only did one step per exercise. For more info see the section on data acquisition.

1	1 1 1 1 1 1
2	1 1
3	0 0 0 1 0 1 1 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 1
4	0 1 1 1 1
5	0 1 1 1 1 1 0 1
6	1 1 1 1 1 1 1
7	0 0 1 1 1 1
8	1 1 1 1 0 0
9	1 1 1 1 1 1
10	0 1 1 1
11	1 1 1 1
12	1 0 1 1 1 1 0 1 1 1 1
13	1 1 1 1
14	1
15	1 1 1 1 1 1 1 1 1 1 1 1
16	0 1 0 0 1 0 1 1 1 1 1 1 1
17	0 1 1 1 1 1 1 1 1
18	0 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1
19	0 0 1 1 1 1 1
20	1 0 1 1 0 1 1 1 0 1
21	1 0 0 1 0 1 1 0 1
22	
23	
24	1 0 1 1 1 1 0 0 1 0 0 1 1 1 1 1 0 0 0 0 0 1 0 0
25	

Table 21: data:(11,17,0), (11,15,1), (11,15,1), (11,21,1), (11,24,1), (11,17,1), (11,6,1), (11,24,0), (11,21,0), (11,5,0), (11,13,1), (11,11,1), (11,12,1), (11,18,0), (11,15,1), (11,3,0), (11,24,1), (11,8,1), (11,11,1), (11,15,1), (11,12,0), (11,18,1), (11,24,1), (11,6,1), (11,7,0), (11,19,0), (11,9,1), (11,13,1), (11,12,1), (11,18,0), (11,14,1), (11,10,0), (11,10,1), (11,24,1), (11,6,1), (11,16,0), (11,18,0), (11,15,1), (11,6,1), (11,1,1), (11,24,1), (11,19,0), (11,18,0), (11,18,1), (11,16,1), (11,3,0), (11,3,0), (11,24,0), (11,8,1), (11,12,1), (11,16,0), (11,15,1), (11,8,1), (11,11,1), (11,12,1), (11,15,1), (11,12,1), (11,17,1), (11,24,0), (11,21,0), (11,3,1), (11,1,1), (11,5,1), (11,9,1), (11,17,1), (11,17,1), (11,7,0), (11,10,1), (11,24,1), (11,18,0), (11,24,0), (11,3,0), (11,8,1), (11,12,0), (11,6,1), (11,20,1), (11,18,1), (11,16,0), (11,24,0), (11,7,1), (11,13,1), (11,12,1), (11,15,1), (11,21,1), (11,5,1), (11,19,1), (11,24,1), (11,20,0), (11,18,1), (11,7,1), (11,10,1), (11,9,1), (11,24,1), (11,17,1), (11,16,1), (11,21,0), (11,7,1), (11,9,1), (11,24,1), (11,17,1), (11,18,1), (11,16,0), (11,15,1), (11,19,1), (11,24,1), (11,18,1), (11,16,1), (11,6,1), (11,13,1), (11,11,1), (11,24,1), (11,12,1), (11,9,1), (11,24,0), (11,3,1), (11,19,1), (11,21,1), (11,3,1), (11,5,1), (11,20,1), (11,20,1), (11,16,1), (11,17,1), (11,1,1), (11,5,1), (11,24,0), (11,3,0), (11,7,1), (11,9,1), (11,3,0), (11,21,1), (11,20,0), (11,3,1), (11,3,1), (11,5,1), (11,18,1), (11,24,0), (11,16,1), (11,4,0), (11,2,1), (11,4,1), (11,3,0), (11,21,0), (11,1,1), (11,3,0), (11,18,1), (11,3,0), (11,8,0), (11,12,1), (11,3,1), (11,19,1), (11,18,1), (11,24,0), (11,3,0), (11,18,1), (11,16,1), (11,15,1), (11,24,0), (11,3,0), (11,1,1), (11,18,1), (11,16,1), (11,15,1), (11,20,1), (11,6,1), (11,24,1), (11,19,1), (11,3,0), (11,24,0), (11,16,1), (11,17,1), (11,3,1), (11,3,0), (11,16,1), (11,15,1), (11,3,0), (11,2,1), (11,4,1), (11,24,0), (11,4,1), (11,3,0), (11,3,1), (11,3,1), (11,3,1), (11,5,0), (11,4,1), (11,21,1), (11,3,1), (11,5,1), (11,20,1), (11,3,0), (11,8,0), (11,12,1), (11,20,1), (11,20,0), (11,3,0), (11,1,1), (11,20,1), (11,3,1)

1	1 1
2	
3	1 0 0 1 1 0 0 1 0 1 1 0 0 1 0
4	0
5	0 1 0 1 0 1 1 0
6	1 1 1 1
7	0 0 0 0 1 0
8	0 0 1 0 1 1 1
9	1 0 1 0 0
10	0 0
11	1
12	1 1 0 0 1 1
13	0 1 1 1
14	1
15	0 1 1 1 0 0 1 1 1 1 1 1
16	0 0 0 1 0 0 0 1 0 1 1 1 1
17	0 0 0 0 0 1 1
18	0 0 0 1 1 0 1 0 0 0 1 0 0 1 1 1 0 1 1 0
19	1 0 0 1 0
20	0 0 0 0 0 0
21	0 0 0
22	
23	
24	0 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
25	

Table 22: data:(12,17,0), (12,24,0), (12,15,0), (12,15,1), (12,20,0), (12,17,0), (12,5,0), (12,18,0), (12,15,1), (12,24,0), (12,3,1), (12,13,0), (12,15,1), (12,8,0), (12,3,0), (12,18,0), (12,18,0), (12,8,0), (12,11,1), (12,18,1), (12,24,1), (12,7,0), (12,6,1), (12,9,1), (12,19,1), (12,24,1), (12,13,1), (12,12,1), (12,18,1), (12,14,1), (12,10,0), (12,24,1), (12,6,1), (12,18,0), (12,15,0), (12,24,1), (12,18,1), (12,19,0), (12,18,0), (12,18,0), (12,16,0), (12,15,0), (12,8,1), (12,24,0), (12,3,0), (12,12,1), (12,15,1), (12,17,0), (12,12,0), (12,24,0), (12,21,0), (12,16,0), (12,20,0), (12,9,0), (12,7,0), (12,24,1), (12,18,0), (12,24,0), (12,8,0), (12,3,1), (12,8,1), (12,18,1), (12,6,1), (12,16,0), (12,20,0), (12,7,0), (12,24,0), (12,13,1), (12,15,1), (12,18,0), (12,7,0), (12,24,0), (12,10,0), (12,9,1), (12,21,0), (12,17,0), (12,7,1), (12,9,0), (12,16,1), (12,18,0), (12,7,0), (12,18,1), (12,24,0), (12,17,0), (12,16,0), (12,15,1), (12,13,1), (12,6,1), (12,18,1), (12,24,0), (12,16,0), (12,9,0), (12,24,0), (12,12,0), (12,3,1), (12,19,0), (12,3,0), (12,5,1), (12,20,0), (12,20,0), (12,16,0), (12,17,1), (12,24,0), (12,5,0), (12,3,0), (12,3,1), (12,5,1), (12,1,1), (12,5,0), (12,1,1), (12,5,1), (12,5,1), (12,18,1), (12,16,1), (12,24,0), (12,3,0), (12,18,0), (12,3,1), (12,8,1), (12,12,1), (12,24,0), (12,21,0), (12,18,1), (12,19,1), (12,3,1), (12,8,1), (12,12,1), (12,18,1), (12,16,0), (12,15,1), (12,24,0), (12,18,0), (12,16,1), (12,15,1), (12,16,1), (12,5,0), (12,20,0), (12,19,0), (12,3,0), (12,4,0), (12,24,0), (12,17,1), (12,16,1), (12,24,0), (12,3,0), (12,3,1), (12,16,1), (12,15,1), (12,3,0)

1	1 1 1 1 1
2	
3	0 0 1 1 1 1 1 1 1 0 1 1 1 0
4	
5	0 1 1 1 1 1 1 1
6	0 1 1 1 1
7	1 1 1 1
8	0 0 1 1 1
9	1 0 1 1 0
10	0 0
11	1 0 0 1
12	1 0 1 1 1 1 1 1
13	0 1 1
14	1
15	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
16	1 1 1 1 1 1 1 1 1 1 1 1 1 1
17	1 1 1 1 1 1 1 0 1 0
18	1 1 1 1 1 1 1 1 1 0 1 1 1 1
19	0 1 1 1 1 1 1 1
20	0 0 0 1 0
21	0 0
22	
23	
24	0 0 1 0 1 1 1 0 1 1 1 1 1 0 1 1 0 0 1 0 1 0 0
25	

Table 23: data:(13,17,1), (13,15,1), (13,15,1), (13,21,0), (13,24,0), (13,17,1), (13,7,1), (13,18,1), (13,15,1), (13,24,0), (13,13,0), (13,11,1), (13,12,1), (13,3,0), (13,15,1), (13,18,1), (13,8,0), (13,24,1), (13,7,1), (13,9,1), (13,6,0), (13,24,0), (13,19,0), (13,18,1), (13,14,1), (13,10,0), (13,13,1), (13,12,0), (13,24,1), (13,6,1), (13,16,1), (13,18,1), (13,17,1), (13,15,1), (13,15,1), (13,1,1), (13,6,1), (13,24,1), (13,18,1), (13,19,1), (13,16,1), (13,15,1), (13,8,0), (13,3,0), (13,8,1), (13,12,1), (13,8,1), (13,11,0), (13,12,1), (13,8,1), (13,11,0), (13,12,1), (13,24,1), (13,15,1), (13,12,1), (13,17,1), (13,3,1), (13,5,0), (13,24,0), (13,17,1), (13,17,1), (13,9,0), (13,24,1), (13,18,1), (13,3,1), (13,24,1), (13,18,1), (13,16,1), (13,15,1), (13,6,1), (13,24,1), (13,15,1), (13,19,1), (13,20,0), (13,5,1), (13,19,1), (13,24,1), (13,18,1), (13,7,1), (13,10,0), (13,9,1), (13,24,1), (13,17,1), (13,16,1), (13,24,0), (13,3,1), (13,5,1), (13,20,0), (13,7,1), (13,9,1), (13,17,0), (13,18,1), (13,16,1), (13,15,1), (13,19,1), (13,24,1), (13,18,0), (13,16,1), (13,6,1), (13,13,1), (13,11,1), (13,12,1), (13,24,1), (13,9,0), (13,12,1), (13,24,0), (13,19,1), (13,16,1), (13,21,0), (13,3,1), (13,5,1), (13,5,1), (13,20,0), (13,1,1), (13,5,1), (13,20,1), (13,17,1), (13,3,1), (13,5,1), (13,18,1), (13,16,1), (13,24,0), (13,3,1), (13,1,1), (13,3,1), (13,19,1), (13,18,1), (13,3,0), (13,24,1), (13,18,1), (13,16,1), (13,15,1), (13,24,0), (13,3,1), (13,18,1), (13,16,1), (13,15,1), (13,16,1), (13,5,1), (13,20,0), (13,24,1), (13,19,1), (13,16,1), (13,17,0), (13,1,1), (13,3,1), (13,24,0), (13,1,1), (13,3,1), (13,16,1), (13,15,1), (13,24,0), (13,3,0)

1	
2	
3	
4	
5	0
6	0 1 0 0 0
7	0 1 0
8	
9	0 0
10	1
11	
12	
13	0 1
14	
15	0 0 1 1 0 1 1
16	0 0
17	0 0
18	0 0 0 0 1 0 0 0
19	0 0 0 0
20	0
21	0 0
22	0
23	
24	1 0 0 0 0 0 1 1 1 1 0
25	

Table 24: data:(14,17,0), (14,5,0), (14,15,0), (14,6,0), (14,22,0), (14,13,0), (14,24,1), (14,6,1), (14,24,0), (14,18,0), (14,15,0), (14,19,0), (14,18,0), (14,24,0), (14,15,1), (14,24,0), (14,17,0), (14,9,0), (14,18,0), (14,24,0), (14,6,0), (14,20,0), (14,18,0), (14,15,1), (14,15,0), (14,18,1), (14,16,0), (14,7,0), (14,13,1), (14,15,1), (14,21,0), (14,15,1), (14,24,0), (14,19,0), (14,18,0), (14,7,1), (14,10,1), (14,21,0), (14,7,0), (14,6,0), (14,24,1), (14,9,0), (14,19,0), (14,24,1), (14,18,0), (14,6,0), (14,18,0), (14,24,1), (14,19,0), (14,24,1), (14,16,0), (14,24,0)

This student used the worked out exercise button quicly when he could not solve the exercise, and remembered the steps. It is impossible to determine which of these steps come from short-term memory and which from competence. Therefore we ignore the rest of the data for the exercise after the worked out solution has been by `ideas`.

C Data in tables per rule (V2)

In this appendix the data is provided in the form of tables of outcomes per rule (V2). Each row contains the sequence of outcomes for one student.

1	0 0 0 0 0 1 0 1 1 1 0 1 1 1 1 0 1 0
2	0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1
3	0 1 0 0 1 1 1 0 0 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1
4	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
5	0 1 0 0 0 0 0 0 1 0 0 0
6	1 0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1
7	1 1 0 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1
8	1 1 1 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 1 0 1
9	0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1
10	
11	0 0 0 1 0 1 1 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 1
12	1 0 0 1 1 0 0 1 0 1 1 0 0 1 0
13	0 0 1 1 1 1 1 1 1 0 1 1 1 0
14	

Table 25: Sequences of outcomes for all students, for rule 3: distribution of and over or $(p \vee q) \wedge r \implies (p \wedge r) \vee (q \wedge r)$.

1	0 0 0 0
2	0 1
3	0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
4	1 1 1 1 1 1 1 1
5	
6	1 1
7	1 1 0 1 1 1 1
8	1 1 1 0 1
9	1 0 1 0 1 1
10	
11	0 1 1 1 1
12	0
13	
14	

Table 26: Sequences of outcomes for all students, for rule 4: distribution of or over and $(p \wedge q) \vee r \implies (p \vee r) \wedge (q \vee r)$.

1	0 1 0 1 1 0 1 1
2	1 1 1 1 0
3	1 1 0 1 1 0 0 1 0
4	0 1 1 1 1 1 1 1 1 1 1
5	0 0 0 0 1 1 1 1 1
6	1 1 1 1 1 1 1
7	0 0 1 1 0 1 1 1
8	1 1 1 1 1
9	1 1 1 1 1 1 1 1 1 1 1 1
10	1
11	0 1 1 1 1 1 0 1
12	0 1 0 1 0 1 1 0
13	0 1 1 1 1 1 1 1
14	0

Table 27: Sequences of outcomes for all students, for rule 5: idempotency of the and $q \wedge q \implies q$.

1	0 1 1 1 1
2	1 1 1 1 1 1 1
3	1 1 1 1 0 1 0 1 1 1
4	1 1 1 1 1 1 1 1
5	1 1 1 1 1 1
6	1 1 1 1 1 1 1
7	1 1 1 1 1 1 1
8	1 1 1 1 1 1
9	1 1 1 1 1 1 1 1
10	0 0 1 0 1 1
11	1 1 1 1 1 1 1
12	1 1 1 1
13	0 1 1 1 1
14	0 1 0 0 0

Table 28: Sequences of outcomes for all students, for rule6: idempotency of the or $q \vee q \implies q$.

1	1 0 0 0
2	1 0 1 0
3	1 1 1 1 1 1 0
4	1 1 0 1 1
5	1 0 1
6	1 1 1 1 1
7	1 0 1 1
8	1 1 1 1 1 1 0 1
9	0 0 0 0
10	
11	0 0 1 1 1 1
12	0 0 0 0 1 0
13	1 1 1 1
14	0 1 0

Table 29: Sequences of outcomes for all students, for rule 7: tautology $p \vee \neg p \implies T$.

1	1 1 0 1
2	0 1 0 0 1 1 1
3	1 1 0 1 0 1 1
4	1 1 1 1 1 1
5	1 0 1
6	1 1 1 1 1 1 1
7	1 1 1 1
8	1 1 0 1 1
9	1 1 1 1 1 1
10	
11	1 1 1 1 0 0
12	0 0 1 0 1 1 1
13	0 0 1 1 1
14	

Table 30: Sequences of outcomes for all students, for rule 8: contradiction $p \wedge \neg p \implies F$.

1	0 0 0 1 1 1
2	1 1 0 1
3	1 1 1 1 1 1 1 1
4	1 1 1 1 1 1
5	0 1 0 0 0 1 0
6	1 1 1 1 1 1
7	1 1 1 1 1
8	0 1 1 1 1 0
9	0 1 1 1 1
10	1
11	1 1 1 1 1 1
12	1 0 1 0 0
13	1 0 1 1 0
14	0 0

Table 31: Sequences of outcomes for all students, for rule 9: $p \wedge T \implies p$.

1	0 1
2	0 0 0
3	1 1 1
4	0 0 0
5	0 0
6	1 1 1
7	1
8	0 0 1 0
9	0 1 0
10	
11	0 1 1 1
12	0 0
13	0 0
14	1

Table 32: Sequences of outcomes for all students, for rule 10: $p \vee T \implies T$.

1	1 1 1 1 1
2	1 0 1
3	1 1 1 1
4	1 1 1 1
5	1 0 0
6	1 1 1
7	1 1 1 1
8	1 1 1
9	1 1 1
10	
11	1 1 1 1
12	1
13	1 0 0 1
14	

Table 33: Sequences of outcomes for all students, for rule 11: $p \wedge F \implies F$..

1	1 1 1 0 1 1 1 1
2	1 1 1 0 1 1 1 1 1
3	1 1 1 1 1 1 1 1 1 1 1 1 1
4	1 1 1 1 1 1 1 1 1 1 1
5	0 0 0 0 0 1 0 1
6	1 1 1 1 1 1 1 1 1 1 1 1
7	0 1 1 1 1 1 1
8	1 1 1 1 1 1 1 1 1 1 1 1
9	0 1 1 1 1 1 1 1 1 1
10	
11	1 0 1 1 1 1 0 1 1 1 1
12	1 1 0 0 1 1
13	1 0 1 1 1 1 1 1
14	

Table 34: Sequences of outcomes for all students, for rule 12: $p \vee F \implies p$.

1	1 1 1 1
2	1 1 1
3	1 1 1 1
4	1 1 1 1
5	1 1 1 1
6	1 1 1
7	1 1 1 1
8	1 1 1 1 1
9	1 0
10	
11	1 1 1 1
12	0 1 1 1
13	0 1 1
14	0 1

Table 35: Sequences of outcomes for all students, for rule 13: $\neg T \implies F$.

1	
2	1
3	1
4	1
5	1
6	1
7	
8	1
9	1
10	
11	1
12	1
13	1
14	

Table 36: Sequences of outcomes for all students, for rule 14: $\neg F \implies T$.

1	0 1 1 1 1 1 1 1 1 1 1
2	1 1 1 1 1 1 1 1 1 1 1 1 1 1
3	1 1 1 1 1 1 1 1 1 1 1 1
4	1 1 1 1 1 1 1 1 1 1 1 1 1
5	0 0 1 0 1 1 1 0 1 1 0 0 0 1 1
6	1 1 1 1 1 1 1 1 1 1 1 1 1
7	1 1 1 1 1 1 1 1 1 1 1 1 1
8	1 1 1 1 1 1 1 0 1 1 1
9	1 1 1 1 1 1 1 1 1 1 1 1 1
10	0 1 1 1
11	1 1 1 1 1 1 1 1 1 1 1 1
12	0 1 1 1 0 0 1 1 1 1 1 1
13	1 1 1 1 1 1 1 1 1 1 1 1 1 1
14	0 0 1 1 0 1 1

Table 37: Sequences of outcomes for all students, for rule 15: double negation $\neg\neg p \implies p$.

1	1 1 1 0 1 1 1 1 1 1 1 1
2	1 0 0 0 1 0 1 0 1
3	0 0 1 0 1 1 1 1 0 1
4	0 1 1 1 1 1 1 1 1 1 1
5	0 0 0 0 0 0 0 0 0
6	1 0 1 1 1 1 1 1 1 1 0
7	1 1 1 1 0 1 1 1 1 1 1 1 0
8	1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1
9	0 1 1 1 0 1 0 0 1 1 1
10	0 0
11	0 1 0 0 1 0 1 1 1 1 1 1 1
12	0 0 0 1 0 0 0 1 0 1 1 1 1
13	1 1 1 1 1 1 1 1 1 1 1 1 1
14	0 0

Table 38: Sequences of outcomes for all students, for rule 16: DeMorgan $\neg(p \vee q) \implies \neg p \wedge \neg q$.

1	0 0 0 0 0 0 1 1 1
2	1 1 0 1 0
3	0 0 1 0 1 1 1 0 0 1 1
4	1 0 1 1 1 1 1 1 0 1 1
5	0 0 0 0 0 0 1
6	1 1 1 1 1 1 1 1 1 1
7	1 1 1 1 1 1 1 1 1 1 1 1
8	0 0 1 1 1 1 1 1 0 1
9	0 1 0 1 1 0 1 1 1 1
10	0 1 0 0
11	0 1 1 1 1 1 1 1 1
12	0 0 0 0 0 0 1 1
13	1 1 1 1 1 1 1 0 1 0
14	0 0

Table 39: Sequences of outcomes for all students, for rule 17: DeMorgan $\neg(p \wedge q) \implies \neg p \vee \neg q$.

1	0 0 0 0 1 0 1 0 0 0 0 1 1 1
2	0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1
3	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
4	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
5	0 0 0 0 0 0 0 0 0 0 0 0 0
6	0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
7	0 1 1 1 1 1 1 1 1 1 1 1 1
8	1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
9	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
10	1 1 1 0 1 1 1
11	0 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1
12	0 0 0 1 1 0 1 0 0 0 1 0 0 1 1 1 0 1 1 0
13	1 1 1 1 1 1 1 1 1 0 1 1 1 1
14	0 0 0 0 1 0 0 0

Table 40: Sequences of outcomes for all students, for rule 18: implication elimination $p \to q \implies \neg p \vee q$.

1	0 0 0 1 0 1 1 0
2	0 1 0 1 1 0 0 1 1
3	0 1 1 1 1 1 1 1
4	1 1 1 0 1 1 1 1 1
5	0 0 1 0 0 0 0
6	0 1 1 1 1 1
7	1 1 1 1 0 1 1
8	0 1 1 1 0 1 1
9	0 0 1 1 0 1 0
10	0 0
11	0 0 1 1 1 1 1
12	1 0 0 1 0
13	0 1 1 1 1 1 1 1
14	0 0 0 0

Table 41: Sequences of outcomes for all students, for rule 19: equivalence elimination $p \leftrightarrow q \implies (p \wedge q) \vee (\neg p \wedge \neg q)$.

1	0 1 1 0 1 0
2	0
3	0 1 0
4	0 0 1 0
5	0 1 0
6	1 0 1 1 1
7	0 0 0 1
8	1 1 1 1 1
9	0 0 0
10	
11	1 0 1 1 0 1 1 1 0 1
12	0 0 0 0 0 0
13	0 0 0 1 0
14	0

Table 42: Sequences of outcomes for all students, for rule 20: absorption $(p \wedge q) \vee q \implies q$.

1	0 1
2	0 0 0
3	0 0 0 0 0 0
4	0 0 1 1 1
5	0 0 0
6	0 1 1 0
7	0 0
8	1 1 1 0 1 1
9	1 0 0 1 1 1 0
10	0
11	1 0 0 1 0 1 1 0 1
12	0 0 0
13	0 0
14	0 0

Table 43: Sequences of outcomes for all students, for rule 21: absorption $(p \vee q) \wedge q \implies q$.

1	0
2	0
3	1
4	
5	
6	
7	1
8	
9	
10	0
11	
12	
13	
14	0

Table 44: Sequences of outcomes for all students, for rule 22: tautology $F \to p \implies T$.

1	1
2	
3	0
4	
5	0
6	
7	1
8	
9	
10	0
11	
12	
13	
14	

Table 45: Sequences of outcomes for all students, for rule 23: tautology $p \to p \implies T$.

1	1 0 1 1 1 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0
2	1 0 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1
3	0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 1 0 1 1 0 0
4	0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 1 1 0 0 0 0
5	0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1
6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
7	0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1
8	1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 1 0 0
9	0 0 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 0 0 1 0
10	
11	1 0 1 1 1 1 0 0 1 0 0 1 1 1 1 1 0 0 0 0 0 1 0 0
12	0 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
13	0 0 1 0 1 1 1 0 1 1 1 1 1 0 1 1 0 0 1 0 1 0 0
14	1 0 0 0 0 0 0 1 1 1 1

Table 46: Sequences of outcomes for all students, for rule 24: ready?.