Dynamic Epistemic Logic Models for Predicting the Cognitive Difficulty of the Deductive Mastermind Game

MSc Thesis *(Afstudeerscriptie)*

written by

Bonan Zhao
(born March 23rd, 1993 in Shandong, China)

under the supervision of **Dr Jakub Szymanik** and **Iris van de Pol**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
August 31, 2017	Prof Benedikt Loewe (chair)
	Dr Peter Hawke
	Dr Lena Kurzen
	MSc Iris van de Pol
	Dr Jakub Szymanik
	Dr Fernando Velazquez

Institute for Logic, Language and Computation

Abstract

This thesis studies the cognitive difficulty of the Deductive Mastermind (DMM) game by measuring the complexity of two different logic formalizations of the game. DMM is a version of the board game Mastermind, and it has been implemented in an online educational game system. This system records players' speed and accuracy data in solving the game, which serves as an empirical indicator of the cognitive difficulty of each DMM game item. In the thesis, we look at an existing formalization of DMM based on analytic tableaux, and we develop a formalization based on dynamic epistemic logic (DEL). The DEL model of DMM performs as well as the tableaux model in predicting the cognitive difficulty of DMM game items, and the DEL model is able to capture more reasoning patterns as self-reported by DMM players. We find that feedback types play an important role in predicting cognitive difficulty of game items, and this result is robust over the two different logic formalizations that we considered.

Acknowledgements

First of all, many thanks toward my dear supervisors Jakub and Iris. Jakub, you are the person that led me to the interdisciplinary study of logic and cognitive science, and thank you for all the meetings, discussions, lectures and conversations we had. Iris, I learned so much from you, from caring about asking *the* question to writing a clear sentence. I would also like to thank Lena, Fernando and Peter for being in my committee, and a special thanks to Peter van Emde Boas for your interest in my thesis project.

My study at the ILLC is possible because of the generous financial support from the Amsterdam Excellence Scholarship, and I am truly grateful to your support. Thank you Fenrong and Johan for introducing logic to me, and supported me to pursue my study in Amsterdam. And thank you Sonja for being my mentor and helping me clear my worries when I got confused or stressed during these two unforgettable years.

Of course, my life at the ILLC is a wonderful experience because of my friends here. The MoL gang gave me the warmest accompany I could ever find, taught me how to appreciate beer and coffee, and showed me the magic of cake and pasta. Amsterdam became a home to me because of you; thank you guys!

Last but not the least, I owe great gratitude to my parents, my partner, and my friends in China. Your trust in me made me through the hard days of the master and will always encourage me in the journey of my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

“Don’t worry,” your friend tells you, “I heard the modal logic final will be easier than our weekly assignments.” In this situation, the difficulty of a logic question is evaluated by how hard it is for a student to solve it. How much time do you spend on it? How directly can you find the right answer? Do you indeed manage to find the right answer in the end? We call this the “cognitive difficulty” of the question or task at hand. We can study the cognitive difficulty of a variety of things such as solving a question in an exam, recognizing a color in a certain context, or interpreting the meaning of a quantifier in communication.

A fruitful method to study the cognitive difficulty of a task is to combine computational and logical analyses. We can use logic to formalize a task into a computational problem, and then measure the complexity of this logic formalization as a predictor for the cognitive difficulty of this task. This two-staged method has proved useful in studying the cognitive difficulty of communicative and linguistic tasks, both theoretically (see Berwick and Weinberg (1984); Cherniak (1986); Barton et al. (1987); Ristad (1993); Szymanik (2016); van Rooij et al. (2011)) and empirically (see Szymanik (2016); Gierasimczuk and Szymanik (2009); Szymanik and Zajenkowski (2010); Zajenkowski et al. (2011)). Feldman (2000) shows that a complexity measure based on logic provides a nice account of the cognitive difficulty of learning various Boolean concepts such as observed in behavioral experiments. Furthermore, this method was successfully applied in studying the compositionality of concept learning (Piantadosi et al., 2016). (See Isaac et al. (2014) for a review on how researchers have applied computational and logical analysis in cognitive science.)

In this thesis we apply this method to a case study. We look at the cognitive difficulty of the Deductive Mastermind (DMM) game. DMM is a simplified version of the board game Mastermind, and to win a DMM game item for a player is to deduce a secret flower sequence, also known as the correct answer, from the information she sees on the screen. Figure 1.1 is a screen shot of a DMM game item. It consists of several clues, each of which includes a sequence of flowers in a line and a corresponding feedback. Besides, it also

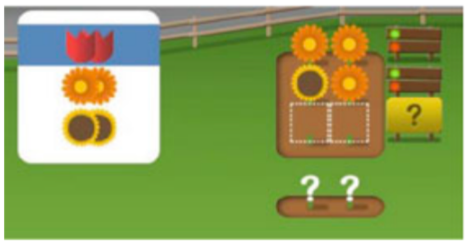


Figure 1.1: A DMM game item

shows some available flowers that a player can choose to form her answer. The feedbacks provide information on the relationship between the flower sequence in a clue and the correct answer. DMM has been implemented in Math Garden, a popular online educational game system, which is used in primary schools all over the Netherlands and has accumulated billions of behavioral data on how children play the games. Each DMM game item in Math Garden is associated with a rating of its difficulty, and this rating is computed based on children's speed and accuracy in solving the game item. DMM is an ideal case study because (1) it can be naturally formalized using logic, and (2) it provides an empirical dataset on the cognitive difficulty of each game item, as indicated by the ratings. By formalizing DMM with logic, we define complexity measures over such formalization, and use these measures as predictors for the cognitive difficulty of DMM game items. Therefore, this is a case study of using logic to capture cognitive difficulty.

Gierasimczuk et al. (2013) propose an analytical tableaux model of DMM that gives an account of the cognitive difficulty of a DMM game item based on the size of the decision tree generated for that item. Their analytic tableaux model correctly predicted 63% of the item ratings,[1] but the model is challenged from different perspectives, which include the following: (1) The tableaux model is based on strong assumptions. The model assumes that players reason by cases and process feedbacks one by one, and Gierasimczuk et al. (2013) assume that the size of tableaux decision trees is a proxy for working memory load. (2) The tableaux model is unable to capture certain observed reasoning patterns. The tableaux formalization is order-dependent. That is, a decision tree in this model can only unfold one clue after another, and therefore cannot represent cross-clue patterns. However, it has been observed that when the clues for a particular game item all contain green-red feedback, children can use this information to make a more strategic move than processing clues one by one (van der Maas, 2017). Consider Figure 1.1 where two green-red feedbacks are given. A child can deduce that the orange daisy that stands at the second place in each clue corresponds to the green feedback and shall go directly to the answer, and the flowers that stand in the first place in the clues correspond to the red feedback peg and therefore should not appear. The tableaux model cannot represent such a move. (3) Since the complexity measurements in the tableaux model are generated by the particular formalization, it is not clear whether they indeed capture the cognitive difficulty of the DMM game, or simply are some artifacts of the formalization itself.

[1]In the 2013 dataset, the tableaux model can predict up to 75% item ratings for 100 game items, and in the 2017 dataset it predicts item ratings at 63% for 355 items.

The above mentioned drawbacks hinder the tableaux formalization, and we want to design a different model that uses fewer assumptions, better represents the choices of the children, and can cross check the plausibility of the logic-based model. We used Dynamic Epistemic Logic (DEL) to build a new model of DMM. This model makes fewer assumptions than the tableaux model, because it does not require reasoning by cases and the way of finding solutions does not depend on the order in which piece of information are processed. The DEL model of DMM solves the game via a natural approach of eliminating impossible options and deliberating over possible answers, and it can give an account of the cross-clue pattern that is mentioned earlier. By testing complexity measurements of the DEL formalization against the empirical dataset, we showed that the DEL model can predict 66% of the item ratings, and thereby performs slightly better than the tableaux model based on the latest dataset. Furthermore, we analyzed the correlation of the DEL and tableaux formalization, and demonstrated that it is the feedback types that determine a game item's cognitive difficulty. This feature is indeed captured by both logic formalizations. These analyses show that the results of the tableaux and the DEL model are not dependent on the particular logic that each model is based on, because they both capture feedback types as predictors of the cognitive difficulty of DMM, irrespective of the the kind of formalization that is being used.

The structure of this thesis is as follows. Chapter 2 introduces the Deductive Mastermind game. It presents both the game setting and the dataset that the game generates. Chapter 3 summarizes the tableaux model. Chapter 4 presents the DEL model, defines several complexity measurements of the DEL model, explains the emergence of cross-clue patterns, and shows how to translate a tableaux decision tree to a DEL model. Chapter 5 tests the complexity measurements of both models with the empirical dataset, and analyzes the results from both formalizations in comparison with each other. Chapter 6 concludes and points out several ideas for future work.

Chapter 2

Deductive Mastermind Game

In this chapter, we introduce the Deductive Mastermind game, both its game setting and the empirical dataset it provides.

2.1 Game Setting

Deductive Mastermind (DMM) is a simplified version of the Mastermind game. Mastermind[2] is a board game between two players, one is called the code-maker and the other is called the code-breaker. The code-maker makes a code that consists of four colored pegs, and places this code below the game board such that the code-maker can see the code, but the code-breaker cannot. Each game consists of several rounds. Each round consists of two parts: first the code-breaker makes a conjecture about the code, then the code-maker gives feedback on the conjecture. There are two types of feedback pegs. A black feedback peg means that a color peg in the code-breaker's conjecture is of the right color and sits in the correct position. A white feedback peg means that a color peg in the code-breaker's conjecture has the correct color but sits in a wrong position. (A feedback of no pegs means none of the colored pegs in the code-breaker's code match a color in the hidden code.) The code-breaker uses the conjectures she has made

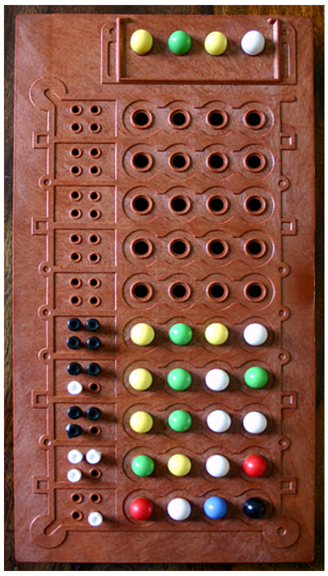


Figure 2.1: A complete Mastermind game won in 5 conjectures

[2]This introduction of Mastermind game is adapted from Mastermind's Wikipedia page (Mastermind (board game), 2017). Readers with a rich knowledge of this game can safely skip this paragraph and go to the next paragraph.

and the feedbacks she has received to formulate a new conjecture for the next round, if the game continues. The code-breaker wins the game if she correctly figures out the secret code within a certain numbers of rounds, and otherwise she loses. Figure 2.1 from Brown (2012) shows a complete Mastermind game where the code-breaker won in five conjectures. In Figure 2.1, the colored code at the upper side of the board is the code-makers secret code, and it is sheltered from the code-breaker. At the bottom side of the board, a code-breaker made five conjectures, and received five feedbacks from the code breaker. The fifth and final conjecture generated an all-black feedback, which meant the code-breaker successfully broke the code.

Mastermind can be turned into a computational problem called Mastermind Satisfiability Problem (MSP): given a set of conjectures and feedbacks, does a unique secret sequence exist that generates the given feedbacks for the given conjectures? Stuckman and Zhang (2005) show that MSP is NP-complete, and they argue that this is why Mastermind has always been a challenging game for human players.

Deductive Mastermind The Deductive Mastermind (DMM) game simplifies the interaction part of a Mastermind game between the code-maker and code-breaker. The computer plays the role of the code-maker, and the human player always plays the role of the code-breaker. The computer displays a set of conjectures with their corresponding feedbacks, and the human player does not formulate his or her own conjectures in the game. The conjectures and feedbacks the computer provide correspond to a unique code, and the task for the human player is to deduce this code.

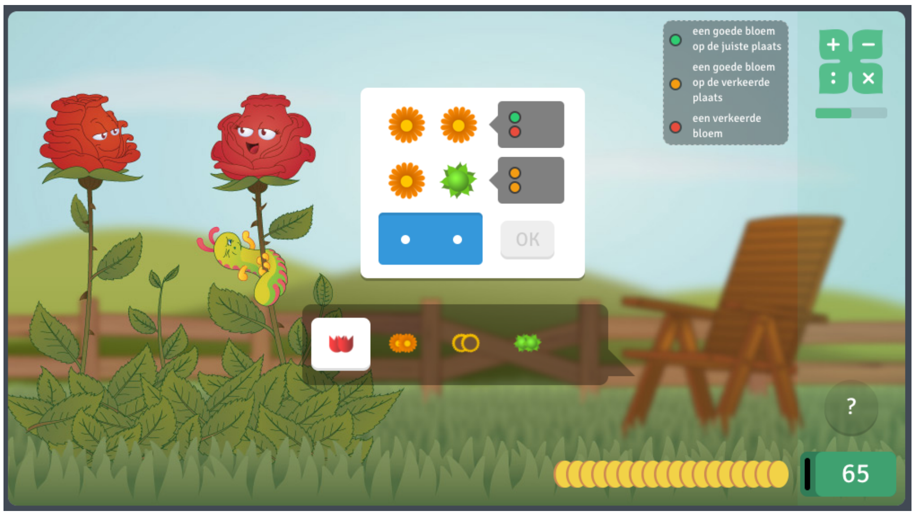

een goede bloem op de juiste plaats
een goede bloem op de verkeerde plaats
een verkeerde bloem
OK
?
65

Figure 2.2: Screen shot of a DMM game item

As mentioned earlier, DMM is implemented in Math Garden, an online educational game system designed for primary school students. In Math Garden, DMM is shown as a game called “Flower Code” among other mathematical games. In this implementation, colored pegs are replaced with flowers, and feedback pegs are presented in more vivid colors, in order to be attractive to children. Throughout this thesis, we write ‘DMM’ to refer to this implementation. Figure 2.2 is a screen shot of a DMM game item. There are 2 clues in this game item, and below the clues there are four types of flower pegs for players to choose from. At the top-right corner, three rules explain what each feedback peg means:

- green peg: a right flower in the right position
- orange peg: a right flower in the wrong position
- red peg: a wrong flower

In DMM, the order in which the feedback pegs are given is always the same: green pegs first, then orange pegs, and lastly red pegs. Hence, no fixed correspondence exists between the positions of flower pegs and the positions of feedback pegs. The first feedback peg is not necessarily meant for the first flower peg in the clue.[3]

In the bottom-right corner, there are some golden coins recording how much time is left for a player to solve the game item. If the player answers correctly, then the less time she used, the more golden coins she will receive as a reward. If she fails to give a correct answer, or she does not answer within the given period of time, then she will not receive any golden coins. This setting serves as a motivation for players to solve the task as quickly and accurately as possible.

We call a DMM game item a n-pin game item if each clue in this game item consists of n flower pegs and n feedback pegs. In 2-pin DMM game items, there are four possible feedbacks: green-red, red-red, orange-orange and orange-red. Other combinations of feedback pegs are not proper feedbacks: green-green simply reveals the game answer and green-orange is not possible.

2.2 Item Ratings

As mentioned earlier, DMM is implemented in Math Garden (`rekentuin.nl` in Dutch, or `MathsGarden.com` in English), an online educational game system where children practice mathematics or analytical skills by playing games. Math Garden was first developed by van der Maas et al. (2010), and by 2013 Math Garden was used in more than 700 primary schools in the Netherlands, and over 90,000 students have generated over 200 billion answers to Math Garden game items (Gierasimczuk et al., 2013).

[3]In a previous version of DMM, feedback pegs were listed as a horizontal line by the side of a conjecture, and researchers found that players tend to correspond the first flower in the conjecture to the first feedback peg, and so on and so forth. Hence, in the new version of this game, feedback pegs are presented vertically as in Figure 2.2, in order to reduce the influence of such correspondence.

Math Garden provides an ideal dataset for studying the cognitive difficulty of playing a game item, because it makes use of a computerized adaptive practice (CAP) system to evaluate a game item's difficulty empirically. This CAP system evaluates a game item's difficulty based on student's speed and accuracy data on solving that item. If more students can solve a game item successfully in a short period of time, then the game item is evaluated as easier, and vice versa. To compute such evaluations, the CAP system extends the Elo rating system (ERS) – a well known interactive rating system that is used to rate chess players – with constraints on time. I will explain how the CAP system works in the following paragraphs, starting from ERS and then introducing constraints on time.

The Elo rating system (ERS) With the purpose of rating chess players, Elo (1978) first developed the Elo rating system (ERS). In ERS, each chess player is rated with a provisional ability rating θ that updates over time according to results of this player's chess matches.

$$\hat{\theta}_j = \theta_j + K(S_j - E(S_j)) \tag{2.1}$$

Equation 2.1 shows how to update player j's rating (denoted as $\hat{\theta}_j$), where S_j is the result of the match for player j. (In chess, S takes the value $0, 0.5$ and 1 for loss, draw and win.) K is a parameter modifying how much one result changes the overall rating, and $E(S)$ is the expected result dependent on ratings of both players in the match. Usually, $E(S_j)$ is calculated as $\frac{1}{1+10^{(\theta_j-\theta_i)/400}}$, where θ_i is the rating of player j's opponent in the match. According to ERS, beating a strong opponent implies that you are also a strong player, and losing to a strong player does not lower your ranking because the system recognizes that the winner is expected to win as a stronger player.

The computerized adaptive practice (CAP) system Math Garden uses the computerized adaptive practice (CAP) system to rate game items, which is an adaptive version of the ERS. In the CAP system, playing a game item is taken as a match between the human player and the computer. If a player solves a game item, it is interpreted as a "win" for the human player, and if the player fails to solve the game item, it is interpreted as a "win" for the computer. In addition, the CAP system also takes time constraints into consideration with the following scoring rule:

$$S_{ij} = (2\chi_{ij} - 1)(a_i d_i - a_i t_{ij}) \tag{2.2}$$

Score S_{ij} is given by a player j's response χ to game item i within time t_{ij}, constrained by time limit d_i and scaled by a discrimination parameter a_i.

The expected score, accordingly, also takes time constraints into consideration, resulting in the following formula:

$$E(S_{ij}) = a_i d_i \frac{e^{2a_i d_i(\theta_j - \beta_i)} + 1}{e^{2a_i d_i(\theta_j - \beta_i)} - 1} - \frac{1}{\theta_j - \beta_i} \tag{2.3}$$

Putting equation 2.2 and equation 2.3 into the original ERS formula 2.1, the Elo rating of a human player θ_j and score of a game item β_i in Math Garden are calculated as follows:

$$\hat{\theta}_j = \theta_j + K_j(S_{ij} - E(S_{ij})) \tag{2.4}$$

$$\hat{\beta}_i = \beta_i + K_i(E(S_{ij}) - S_{ij})) \tag{2.5}$$

Note that we use the term "Elo rating" or "rating" to refer to the rating computed by the CAP system from now on. These equations show that if more players are able to solve a game item fast and correctly, then the game item is evaluated with a lower rating, and if few players can solve a game item correctly, then this game item is evaluated with a higher rating. Players' performance data, specifically time and accuracy, provide an empirical measurement for how difficult a game item is behavior-wise. In Math Garden, Elo ratings of human players and game items range from $-\infty$ to $+\infty$. In general, the lower the Elo rating, the easier a game item is, and the higher the Elo rating is, the harder a game item is. (For more details, readers can consult Klinkenberg et al. (2011) and Maris and van der Maas (2012).)

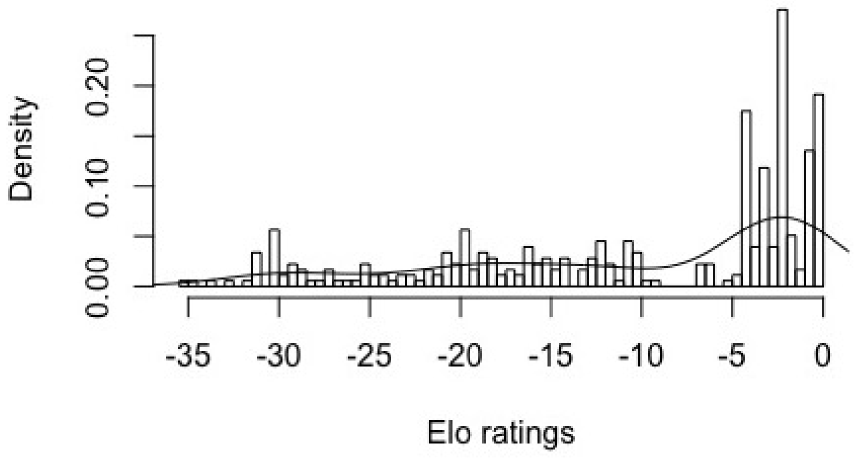


Figure 2.3: Ratings of 2-pin DMM game items

Dataset for 2-pin DMM game items In this thesis, we look at the dataset provided by Math Garden of item ratings for 2-pin DDM game items. In practice, among all DMM game items, 2-pin items are played most often, and thus their ratings are more reliable.

Up to May 2017, there are 355 2-pin DMM game items in Math Garden, and their Elo ratings range from -35.23118 to -0.001269599. Elo ratings for 2-pin game items are negative mainly because most players are able to solve 2-pin game items successfully. Figure 2.3 shows the distribution of Elo ratings for 2-pin DMM game items. The x-axis shows Elo ratings for 2-pin DMM game items, and the y-axis shows the frequency of the Elo ratings. This plot shows that the distribution of the Elo ratings of 2-pin DMM has a peak around -2.5, while it is more uniformly distributed for game items with Elo ratings below -10.

Chapter 3

Analytical Tableaux Model

As described in the previous chapter, in a DMM game item players need to deduce the secret flower code from a set of clues displayed on the screen. This makes DMM a game that can be naturally formalized in logic. In addition, the CAP system used in Math Garden assigns each DMM game item an Elo rating that reflects the cognitive difficulty of that item based on the accumulated data. Therefore, a logic based complexity analysis can be applied to study the cognitive difficulty of playing DMM.

Gierasimczuk et al. (2013) were the first to study the cognitive difficulty of playing DMM game items on the basis of a logic formalization of the game. They proposed an analytical tableaux model for 2-pin DMM game items. Analytical tableaux is a decision procedure for finding a satisfying valuation for a given propositional formula (Beth, 1955; Smullyan, 1968; van Benthem, 1974). Gierasimczuk et al. (2013) first converted each 2-pin DMM game item into a set of Boolean formulae, and then built a decision tree for each game item following the analytical tableaux method. The assumption was that the size of the decision tree is a proxy of working memory load for solving this game item, and linear regression analysis showed that the size of the decision trees predicted 75% the ratings of the game items correctly.

In this chapter, we present the formalization of 2-pin DMM game items in the tableaux method used in Gierasimczuk et al. (2013), and analyze the virtues and shortcomings of this model.

3.1 Formalization

The analytical tableaux model of 2-pin DMM game items views each game item as a set of Boolean formulae, and solving the game item is equivalent to finding the unique valuation that satisfies these formulas. To that end, a conjecture in a game item is viewed as an assignment of flowers. Formally,

Definition 3.1 (Conjecture)**.** A conjecture of length l (consisting of l pins) over k flowers is defined as a sequence given by a total assignment $h : \{1, \dots, l\} \to \{c_1, \dots, c_k\}$. In the game setting, the goal sequence *goal* is a specific conjecture, $goal : \{1, \dots, l\} \to \{c_1, \dots, c_k\}$.

According to this definition, $h(i), goal(j), i, j \in \{1, \dots, l\}$ refer to flower pegs, and $h(i) = goal(j)$ where $i, j \in \{1, \dots, l\}$ is viewed as a literal in the Boolean translation of a game item.

Every non-goal conjecture is paired with a feedback that indicates how similar h is to the given *goal* assignment. There are three types of feedback pegs: green, orange, and red. In the model, green is represented by g, orange by o and red by r.

Definition 3.2 (Feedback)**.** The feedback f for flower configuration h with respect to *goal* is a sequence

$$\overbrace{g \dots g}^{a}\overbrace{o \dots o}^{b}\overbrace{r \dots r}^{c} = g^a o^b r^c$$

where $a, b, c \in \{0, 1, 2, 3, \dots\}$ and $a + b + c = l$.

A feedback consists of

- exactly one g for each $i \in G$ where $G = \{i \in \{1, \dots l\} \mid h(i) = goal(i)\}$,
- exactly one o for every $i \in O$, where $O = \{i \in \{1, \dots, l\} \setminus G|$ there is a $j \in \{1, \dots, l\} \setminus G$, s. t. $i \neq j$ and $h(i) = goal(j)\}$, and
- exactly one r for every $i \in \{1, \dots, l\} \setminus (G \cup O)$.

Sets G, O, and R induce a partition over $\{1, \dots, l\}$, and Gierasimczuk et al. (2013) define $\varphi^g_G, \varphi^r_{G,o}, \varphi^o_{G,O}$ to represent the the propositional formulae that correspond to different parts of the feedback.

- $\varphi^g_G := \bigwedge_{i \in G} h(i) = goal(i) \wedge \bigwedge_{j \in \{1,\dots,l\} \setminus G} h(j) \neq goal(j)$
- $\varphi^o_{G,O} := \bigwedge_{i \in O} (\bigvee_{j \in \{1,\dots,l\} \setminus G, i \neq j} h(i) = goal(j))$
- $\varphi^r_{G,O} := \bigwedge_{i \in \{1,\dots,l\} \setminus \{G \cup O\}, j \in \{1,\dots,l\} \setminus G, i \neq j} h(i) \neq goal(j)$

Gierasimczuk et al. (2013) then set $\mathbb{G} := \{G | G \subseteq \{1, \dots, l\} \wedge card(G) = a\}$ and if $G \subseteq \{1, \dots, l\}$,then $\mathbb{O}^G = \{O | O \subseteq \{1, \dots, l\} \setminus G \wedge card(O) = b\}$. These two sets are collections of possible assignments with respect to a specific feedback. With the help of these sets, a clue can be translated into a Boolean formula as follows:

Definition 3.3 (Boolean translation of a clue)**.** The Boolean translation of a clue consisting of conjecture h with its corresponding feedback f is given by

$$Bt(h, f) := \bigvee_{G \in \mathbb{G}} \left(\varphi^g_G \wedge \bigvee_{O \in \mathbb{O}^G} (\varphi^o_{G,O} \wedge \varphi^r_{G,O}) \right).$$

Putting clues together, a game item is then viewed as a set of Boolean formulae.

Definition 3.4 (Boolean translation of an item)**.** A DMM game item over l pins, k flowers and n rows, $DM(l, k, n)$, is a set of clues $\{(h_1, f_1), \ldots, (h_n, f_n)\}$, each consisting of a single conjecture h_i and its corresponding feedback f_i. The Boolean translation of a DMM-item $Bt(DM(l, k, n)) = Bt(\{(h_1, f_1), \ldots, (h_n, f_n)\}) = \{Bt(h_1, f_1), \ldots, Bt(h_n, f_n)\}$.

Let us look at Example 3.1 to understand the translation of a DMM game item in the tableaux formalization.

Example 3.1. Consider the game item in Figure 2.2. For the first row of conjecture, take (h_1, f_1) such that $h_1(1) := c_2, h_1(2) := c_2$, and the corresponding feedback $f_1 := gr$, then $\mathbb{G} = \{\{1\}, \{2\}\}, \mathbb{O}^G = \varnothing$, and

$$Bt(h_1, f_1) = (goal(1) = c_2 \wedge goal(2) \neq c_2) \vee (goal(2) = c_2 \wedge goal(1) \neq c_2).$$

Similarly, for the second row of conjecture (h_2, f_2) such that $h_2(1) := c_2, h_2(2) := c_4$ and feedback $f_2 := oo$, $\mathbb{G} = \{\varnothing\}$ and $\mathbb{O}^G = \{\{1, 2\}\}$. Hence,

$$Bt(h_2, f_2) = goal(1) \neq c_2 \wedge goal(2) \neq c_4 \wedge goal(1) = c_4 \wedge goal(2) = c_2.$$

The Boolean translation for all clues in 2-pin DMM game items are listed in Table 3.1. For any $h(1) := c_i, h(2) := c_j, Bt(h, f)$ is:

Feedback f	Boolean Translation $Bt(h, f)$
oo	$goal(1) \neq c_i \wedge goal(2) \neq c_j \wedge goal(1) = c_j \wedge goal(2) = c_i$
rr	$goal(1) \neq c_i \wedge goal(1) \neq c_j \wedge goal(2) \neq c_i \wedge goal(2) \neq c_j$
gr	$(goal(1) = c_i \wedge goal(2) \neq c_j) \vee (goal(2) = c_j \wedge goal(1) \neq c_i)$
or	$(goal(1) \neq c_i \wedge goal(2) \neq c_j) \wedge$ $(goal(1) = c_j \wedge goal(2) \neq c_i) \vee (goal(2) = c_i \wedge goal(1) \neq c_j)$

Table 3.1: Boolean translations for 2-pin DMM feedbacks

After translating each 2-pin DMM game item into a set of Boolean formulae $DM(l, k, n)$, the analytical tableaux method is applied to build a decision tree for a game item in order to find the unique valuation *goal*. In analytical tableaux there are standard rules of unfolding a Boolean formula into a decision tree. In the case of 2-pin DMM game items, only two logical connectives are used, namely, $\wedge$ and $\vee$, as listed in Table 3.1, because negation only takes place at the literal level. Hence, there are four branching rules for formulae of 2-pin DMM game items, and they are depicted in Figure 3.1. Figure 3.2a shows the decision tree for the game item in Example 3.1 following the top-to-bottom order.

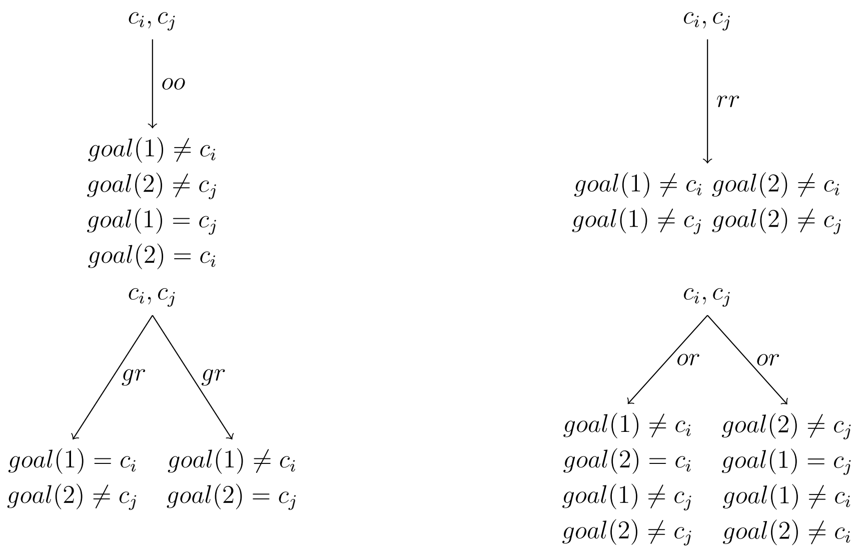


Figure 3.1: Branching rules for 2-pin DMM feedbacks

3.2 Complexity Measurements

Gierasimczuk et al. (2013) define the size of a decision tree to be the measurement of complexity for a decision tree. Given a logic formalization, a complexity measure over that formalization is a formal notion that captures some combinatorial property of the formalization. The goal of such measurements is to investigate whether this formal property captures some of what causes the cognitive difficulty of the task. Since the decision tree is viewed as how children find solutions for a DMM game item, the size of the decision tree is therefore viewed as a proxy of working memory load that predicts the Elo rating of a game item (Gierasimczuk et al., 2013). Observing the trees in Figure 3.1, it is obvious that different feedbacks result in different branching and different sizes of decision trees. Ordering by the size of decision trees generated by each feedback, the tree-difficulty for the four types of feedbacks in 2-pin DMM game items is: $oo < rr < gr < or$.

Obviously, processing the easier feedbacks earlier can shrink the size of the decision tree, while processing the harder feedbacks earlier will amplify the size of the decision tree. For example, Figure 3.2 shows two different decision trees for the DMM game item in Figure 2.2. The left tree is built according to the default order of conjectures, i.e., first

$Bt(h_1, f_1)$ and then $Bt(h_2, f_2)$. Since $f_1 = gr$, the tree branches at the first level, and with $f_2 = oo$, each of the branches extends one step further, resulting in a decision tree with four branches. On the right tree in Figure 3.2, however, if the agent starts building the tree from $Bt(h_2, f_2)$ directly, since $f_2 = oo$, by moving one step the agent can already find a valuation that satisfies the Boolean formulae of this game item, and this decision tree has just one branch.

$$\begin{tikzcd}
& Bt(h_1, f_1) \arrow[dl, "gr"] \arrow[dr, "gr"'] & \\
\begin{array}{c} goal(1) = c_2 \\ goal(2) \neq c_2 \\ Bt(h_2, f_2) \end{array} \arrow[d, "oo"] & & \begin{array}{c} goal(1) \neq c_2 \\ goal(2) = c_2 \\ Bt(h_2, f_2) \end{array} \arrow[d, "oo"] \\
\begin{array}{c} goal(1) = c_4 \\ goal(2) = c_2 \end{array} & & \begin{array}{c} goal(1) = c_4 \\ goal(2) = c_2 \end{array}
\end{tikzcd}$$

(a) The default decision tree

$$\begin{tikzcd}
Bt(h_2, f_2) \arrow[d, "oo"] \\
\begin{array}{c} goal(1) = c_4 \\ goal(2) = c_2 \\ Bt(h_1, f_1) \end{array}
\end{tikzcd}$$

(b) The least decision tree

Figure 3.2: Two different decision trees for the same game item

Therefore, Gierasimczuk et al. (2013) proposed two ways of solving a 2-pin DMM game item. One is to process feedbacks from top to bottom, generating a *default* decision tree, and another is to process feedbacks following the difficulty order $oo < rr < gr < or$, generating the *least* decision tree. Note that not all decision trees lead to *goal* valuation directly. In some cases, a flower is not used in formulating clues, and one needs to add that flower to the final valuation in order to produce the correct answer.

Application steps After building the decision trees, the next step is to measure the size of a decision tree as the indicator for item ratings. In the analytical tableaux method, Gierasimczuk et al. (2013) used application steps per feedback of a decision tree to measure the size of decision trees. Application steps are computed by a recursive algorithm that assumes that agents search over the tree following the top-to-bottom and left-to-right order, and once a solution is found, the algorithm will stop. If the search meets a contradiction at some node, then it goes one step back and continues. For each feedback that appears in the game item, the application steps for that feedback is the number of searches this algorithm conducts on the branches generated by that feedback. If one of the four feedbacks does not appear in a game item, the application step of that feedback is set to 0. If a feedback appears more than once, then the application steps of this feedback is the sum of all application steps the feedback generates at different level of the tree.

Example 3.2. Consider a DMM game item $DM(l = 2, k = 3, n = 2)$. This item has 2 pins, 3 types of flowers, and 2 rows. Let $h_1(1) := c_1$, $h_1(2) := c_2$, $f_1 := gr$ and $h_2(1) := c_3$, $h_2(2) := c_2$, $f_2 := gr$. The default tree and least tree for this game item are the same, which is depicted in Figure 3.3. Number of steps are counted at each gr-edge, and for this game item the application steps for gr is 6, application steps for oo, rr and or are all 0.

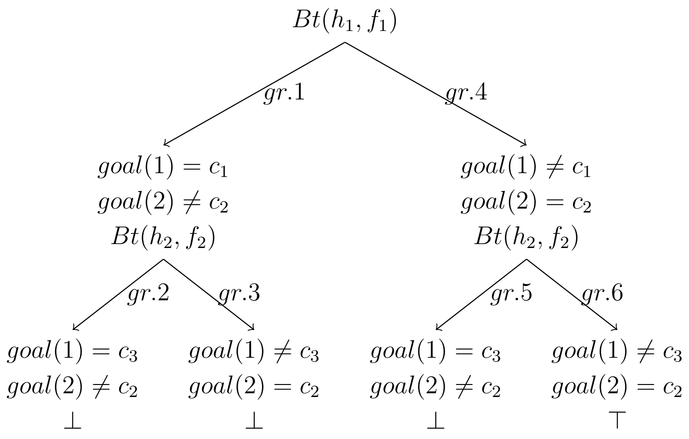


Figure 3.3: Tableaux tree for Example 3.2

Regression results The application steps are treated as the size of the decision tree in Gierasimczuk et al. (2013), and are used as the complexity measurement of the tableaux model of DMM. The application steps of a DMM game item is a tuple of four, and each element in the tuple represents the application steps for a feedback type. Gierasimczuk et al. (2013) tested how well application steps predicted item ratings on 100 2-pin DMM game items, and the results showed high correlation. A basic regression model that only considered basic game features such as number of flower types, number of clues, and whether all flowers were used in the clues, could only explain 34%[4] of the variance. However, the regression model that incorporated application steps based on the default decision tree could explain 70% of the variance, and the regression model that incorporated application steps based on the least decision tree could explain 75% of the variance. This showed that application steps computed by the analytical tableaux model can predict item ratings very well, and the size of the decision trees is a possible explanation for the cognitive difficulty of 2-pin DMM game items.

[4]Results in Gierasimczuk et al. (2013) were generated based on the 2013 dataset of DMM. In Chapter 5 Empirical Evaluation of Models, we tested the tableaux model with the latest dataset, and replicated similar results. A more detailed statistical analysis can be found in Chapter 5.

3.3 Caveats

Even though the analytical tableaux model is able to correctly predict the the Elo ratings of game items, some of the assumptions and limitations of this model limit its ability to adequately explain the cognitive difficulty of DMM game items.

Order-dependency The procedure for building a decision tree in the analytical tableaux model is order dependent. According to the rules of analytical tableaux, a decision tree has to be unfolded step by step. In the case of DMM, a step is a Boolean formula that represents a clue. Therefore, in a DMM game, the analytical tableaux model depicts an agent as reasoning about clues one by one. Hence, the analytical tableaux model makes strong assumptions on the order in which players reason about clues when working on a game item.

However, researchers have observed that in reality players also reason across clues. When children saw game items whose feedbacks are all *gr* and the same flower appeared at the same position in the clues, they chose that flower and put it in the same position as in the clues for their answer. Self-reports of children supported that some children did recognize this pattern of all *gr* feedbacks. By applying some logic reasoning, clues that all contain *gr* feedbacks form a logical shortcut for solving that game item. Researchers also speculated that children are even able to use this pattern in game items with more pins. Figure 3.4 gives an example of an all *gr* game item. In these kinds of cases the analytical tableaux model provides an incorrect prediction by stating this item is very difficult because it generates a huge decision tree that branches four times, whereas this item is, in fact, easy because children can recognize the all *gr* pattern across clues.

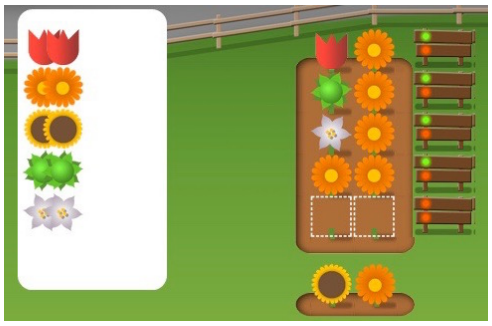


Figure 3.4: Game item with four all *gr* feedbacks

Reasoning by cases The analytical tableaux model makes strong assumptions about the cognitive process of children when they play this game. Gierasimczuk et al. (2013) claim that a decision tree generated by the tableau method for a DMM game item "represents an adequate reasoning scheme" for players. In the analytical tableaux model, the size of the decision tree is determined by the branching rules for feedbacks. Feedbacks such as *gr* and *or* branch the decision tree, and therefore increase the complexity of the resulting decision tree. Reasoning by case directly decides the complexity measurements. To put it bluntly, the tableaux model assumes that it is reasoning by cases that decides the cognitive difficulty of a 2-pin DMM game item.

Even though the tableaux model implicitly assumes that decision trees are how children solve a game item, in practice uncovering the actual reasoning procedure that children use is quite difficult. Instead of reasoning by cases, children may also think in different ways, such as deliberating over possible answers and eliminating impossible conjectures. Therefore, it is too strong to assume that decision trees generated by tableaux is *the* cognitive process for solving that game.

Reliability of parameters Another concern follows naturally from the worry about making strong assumptions about the cognitive process is that, since the size of decision trees is a feature of the formal system, it is not clear whether the application steps indeed capture the cognitive difficulty of a game item, or just are artifacts of the specific model being used. Parameters computed by a particular formalization reflect properties of a game item under that formalization. But if that formalization is not a good representation of our cognitive system, to what extent can we take those formal parameters to be parameters about a game item's cognitive difficulty?

Chapter 4

Dynamic Epistemic Logic Model

We saw an analytical tableaux model for 2-pin DMM game items. This model was able to correctly predict 75% of the ratings of game items, but is also challenged for claiming to represent the cognitive process of solving DMM games without accounting for observed reasoning patterns. Since children may solve DMM games using processes other than reasoning by cases, can some other model capture those processes? Can we cross-check the reliability of the tableaux model by another model based on a different formalization?

The answer to each of these questions is yes. In this chapter we explore a Dynamic Epistemic Logic (DEL) model of 2-pin DMM game items. The DEL model of DMM game items does not include an assumption of reasoning by cases, and it allows agents other ways of solving a 2-pin DMM game item such as by deliberating over possible options. The DEL model can represent both order-dependent and order-independent ways of solving a game item, and provides a nice representation of cross clue logical shortcuts like the all *gr* feedback pattern discussed earlier. In addition to these benefits, the DEL model of DMM game items is at least as general as the tableaux model because each tableaux decision tree for a DMM game item can be translated into a DEL model of the same game item.

In this chapter, we present the DEL formalization of DMM game items and the DEL way of solving a 2-pin DMM game item. We present two variants of the DEL model. One is order-dependent, and the other order-independent. We discuss and define complexity measurements of the DEL models, and show how to translate a tableaux decision tree into a DEL model, and end this chapter by presenting the DEL account for cross clue logical shortcuts.

4.1 DEL Preliminaries

This section refreshes readers on the knowledge of DEL required for the modeling. The definitions we introduce here are primarily based on lecture notes from Baltag (2016).

As a unifying framework of both epistemics and dynamics, DEL extends basic epistemic logic with event models and product updates, and thus becomes a powerful framework that can model sophisticated belief revision, information flow in social interactions, and many other phenomena (Baltag et al., 1998; van Ditmarsch et al., 2007; van Benthem et al., 2006). The basic language of DEL is the same as standard epistemic logic.

Definition 4.1 (Language)**.** The language of of single-agent epistemic logic $\mathcal{L}_{\mathcal{X}}$ is generated by:

$$\varphi \ ::= \quad p \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \varphi \quad | \quad B\varphi$$

where $p \in \Phi$ and Φ is a countable set of atomic sentences. $\vee$ and $\rightarrow$ are defined in the standard way. $B\varphi$ reads like "believe φ".[5]

In DEL, the epistemic states of agents are represented by epistemic models.

Definition 4.2 (Epistemic model)**.** An epistemic model of $\mathcal{L}_{\mathcal{X}}$ is a tuple $\mathbf{S} = \langle S, ||\cdot||, s^* \rangle$ where S is a set of possible worlds that are epistemically possible, $||\cdot|| : \Phi \mapsto \mathcal{P}(S)$ is a valuation assigning to each $p \in \Phi$ a set $||p||_S$ of worlds, and s^* is the actual world.

This model provides sphere semantics for $\mathcal{L}_{\mathcal{X}}$, which differs from the standard Kripke semantics for epistemic models, by not having indistinguishablity relations. Therefore, possible worlds are not connected in this epistemic model; instead, they are evaluated as sets. The sphere models are less general than Kripke models for $\mathcal{L}_{\mathcal{X}}$, but is enough for modeling DMM game items and solutions. For any world w in a model $\mathbf{S}$ and any sentence φ, we write $w \models_{\mathbf{S}} \varphi$ if φ is true in the world w. When the model $\mathbf{S}$ is fixed, we omit the subscript and simply write $w \models \varphi$. For atomic sentences, $w \models \varphi$ is given by the valuation:

$$w \models p \ \text{ iff } \ w \in ||p||$$

The semantics for other propositional formulas is given by the usual truth clauses:

$$\begin{aligned} w \models \neg\varphi \ &\text{ iff } \ w \not\models \varphi \\ w \models \varphi \wedge \psi \ &\text{ iff } \ w \models \varphi \text{ and } w \models \psi \end{aligned}$$

The semantics for the belief operator B is given by:

$$w \models B\varphi \ \text{ iff } \ t \models \varphi \text{ for all } t \in S$$

Definition 4.3 (Event model)**.** An event model is $\mathbf{E} = \{E, pre\}$,[6] where $e \in E$ is an action, and pre is a sentence in $\mathcal{L}_{\mathcal{X}}$ that describes the precondition of E.

[5]In the standard epistemic logic, there is another modal operator $K\varphi$ that reads "know φ". For the scope of this thesis, the belief operator B is enough for our modeling, and therefore we omit this operator.

[6]We omit the indistinguishablity relation ($\xrightarrow{A}$) here because: (a) this is a single-agent model, so we do not need to specify agents, and (b) sphere semantics does not involve indistinguishablity relations. Same for the product update model.

Given epistemic models and event models defined above, agents update their beliefs according to the product update protocol:

Definition 4.4 (Product update)**.** The product update model is defined as

$$\mathbf{S} \otimes \mathbf{E} = (S \otimes E, || \cdot ||)$$

where $\{(s, e) \in S \otimes E \mid s \models pre_e\}$ and $||p||_{\mathbf{S} \otimes \mathbf{E}} := s \in ||p||_{\mathbf{S}}$.

4.2 Model of Game Items

In this section, we present the DEL formalization of DMM game items and how to solve a DMM game item via product update in DEL. This section divides into two subsections. The first subsection formalizes static information of a DMM game item shown on the screen. The second shows how to use DEL techniques to model the process of finding the secret flower code.

4.2.1 Formalizing Game Items

We formalize the static information shown on the screen for a DMM game item in this subsection, and we call this formalization the DMM game model. The DMM game model is introduced in three steps, first we show the mapping of flower pegs and feedback pegs to propositional letter, then we introduce sentences that represent flower sequences and clues, and lastly we define a feedback function that encodes how feedback pegs are given with respect to the secret sequence and a flower sequence.

Atomic sentences

The set of atomic sentences Φ consists of propositional letters for flower pegs and feedback pegs, as well as indexed propositional letters for flower pegs in flower sequences. We assign propositional letters as follows: $p, q, \ldots$ to lower pegs, g to green feedback pegs, o to orange feedback pegs, and r to red feedback pegs. For flower pegs in a flower sequence, index $\cdot_i$ denotes the position of each flower in the sequence.

Example 4.1. As an example, consider the DMM game item in Figure 2.2. Let a stand for tulips, b stand for daisies, c stand for sunflowers and d stand for the green flowers, b_1 stand for a daisy that appears at the first place in a flower sequence, b_2 stand for a daisy that appears at the second place in a flower sequence, and d_2 stand for a green flower that appears at the second place in a flower sequence. The first row of the flower sequence in Figure 2.2 consists of propositions b_1 and b_2, and the second row of the flower sequence consists of propositions b_1 and d_2.

We can concatenate n propositional letters for feedback pegs to form a feedback sequence σ^n. For example, a 2-pin DMM game items has four possible feedbacks: green-red, red-red, orange-orange and orange-red. Hence, the four possible feedback sequences σ^2 are gr, rr, oo, and or. When $n = 1$, the three possible feedback sequences σ^1 are the three propositional letters for feedback pegs, namely, g, r, and o. We include feedback sequences $\sigma^n, n \geq 1$ in the set of atomic sentences Φ as well.

Altogether, the set of propositional letters Φ consists of flower pegs $p, q, \ldots$, indexed flower pegs $p_i, q_j, \ldots$, and feedback sequences $\sigma^n, n \geq 1$.

DMM game model

For every DMM game item, each clue consists of a conjecture and a corresponding feedback. The conjecture consists of n flower pegs and the feedback consists of n feedback pegs. Formally, we define a clue as follows.

Definition 4.5 (Clue)**.** A clue $L = \bigwedge_{i=1}^{n} x_i \wedge \sigma^n, \quad x, x_i, \sigma^n \in \Phi$.

In Definition 4.5, the x of x_i ranges over propositional letters for flowers $p, q, \ldots$, and i is the position of the flower in the clue. The feedback sequence σ^n in the clue has length n, which equals the number of indexed flower pegs. Note that σ^n is a propositional letter in our language. We call the conjunction of all the x_is in a clue L a *flower configuration*, and denote it with C. A clue is a conjunction of a flower configuration $C = \bigwedge_{i=1}^{n} x_i$ and a feedback sequence σ^n. For every clue, each position in the flower configuration shall be filled with exactly one flower. In other words, for all $1 \leq i \leq n$, there exists exactly one x_j such that $x_i = x_j$. If p_i, q_j are in the same clue and $i = j$, then $p = q$; but not the converse.

Example 4.2. Figure 2.2 shows 2 clues of a game item. In the first row clue $L_1 = b_1 \wedge b_2 \wedge gr$, and in the second row clue $L_2 = b_1 \wedge d_2 \wedge oo$.

The secret code, or *goal* (the only correct answer of the game), is a flower configuration that is hidden from agents in the game.

We now define the game model for DMM game items.

Definition 4.6 (DMM game model)**.** A DMM game model G is a tuple $\langle F_G, \mathcal{L}_G, goal_G \rangle$ where F_G is a set of indexed flower pegs that are available for players to choose from, $\mathcal{L}_G$ is the set of clues, and $goal_G$ is the secret code.

The secret code *goal* is not shown on the screen for players, but we still include it determines how feedbacks are given for a flower configuration. In the next step, we define a feedback function that interprets the relation between flower sequences and feedbacks with respect to *goal*. For simplicity, we omit the subscript $\cdot_G$ when the context is clear.

Example 4.3. For example, consider again the game item in Figure 2.2. Let G be the DMM game model of this game item, $F = \{a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2\}$, and $\mathcal{L} = \{L_1, L_2\}$. As in Example 4.2, clue $L_1 = b_1 \wedge b_2 \wedge gr$, and clue $L_2 = b_1 \wedge d_2 \wedge oo$. Note that $goal$ is not on the screen.

Feedback function

We want to not only describe DMM game items in DEL, but also encode the information needed for an agent to solve those game items. At the top-right corner of the screen shot in Figure 2.2, 3 rules state the relation between feedback pegs, conjectures, and the secret code. Accordingly, we define the following function to represent these pieces of information.

Definition 4.7. Given a DMM game model $G = \langle F, \mathcal{L}, goal \rangle$, a clue $L \in \mathcal{L}$ and all the p_i that appear in L:

$$f^{goal}(p_i) = \begin{cases} g & \text{if there is a } q_j \text{ in } goal \text{ such that } p = q \text{ and } i = j; \\ o & \text{there is a } q_j \text{ in } goal \text{ such that } p = q \text{ and } i \neq k \\ & \text{and there is no } r_k \text{ in } L \text{ such that } q = r \text{ and } j = k; \\ r & \text{otherwise.} \end{cases}$$

Let C be the flower configuration in clue L, with a slight abuse of notation, we write $f^{goal}(C) = \sigma^n$ as the result of concatenating $f^{goal}(p_i)$ for all p_i that appears in C. Therefore, a clue $L = C \wedge \sigma^n = C \wedge f^{goal}(C)$.

Example 4.4. Take game model G in Example 4.3, it is the case that $f^{goal}(C_1) = f^{goal}(b_1 \wedge b_2) = gr$ and $f^{goal}(C_2) = f^{goal}(b_1 \wedge d_2) = oo$.

The feedback function specifies the relation between a flower configuration C in clue L and the $goal$ configuration. It encodes information for solving a DMM game item. This definition can be relaxed to any flower configuration instead of just $goal$, which we will use later.

4.2.2 Solving Game Items

Now that we have formalized DMM game items into DMM game models, we can build DEL models for each game model to analyze the procedure of solving the game.

In general, an epistemic model describes an agent's beliefs about possible answers to the game. An event model represents the action of reasoning about a clue. The product model that results from updating an epistemic model with an event model forms a new epistemic model for an agent after reasoning about the clue.

Taking the framework of DEL, we get the syntax and semantics for epistemic model, event model and product update for free. What remains to be specified are the valuation function and preconditions. Let us examine them one by one.

Valuation function

In an epistemic model $\mathbf{S} = \{S, ||\cdot||, s^*\}$, a possible world $s \in S$ such that $s \models \varphi$ represents a possibility that φ is true at s. As for modeling DMM game items, we want each possible world s to represent a possible flower configuration. In other words, for each possible flower configuration C, we want one possible world $s \in S$ such that $s \models C$.

Recall that a flower configuration C is a conjunction of indexed flower pegs, $\bigwedge_{i=1}^{n} x_i$. Hence, by the semantics of DEL, we want the valuation function $||\cdot||$ to map the set of indexed flower pegs F_G to the power set of possible worlds $\mathcal{P}(S)$, i.e., $||\cdot|| : F_G \mapsto \mathcal{P}(S)$. The actual world $s^* \models goal$. Atomic sentences that are not in the set of indexed flower pegs are mapped to the empty set, i.e., $||\cdot|| : \Phi \setminus F_G \mapsto \varnothing$.

Before an agent reasons about any clue, every flower configuration is a possible answer for the game. We call the epistemic model before updating with any event models the initial epistemic model. Hence, the number of possible worlds in the initial epistemic model is the same as the number of all possible flower configurations.

Proposition 4.1. Let $\mathbf{S} = \{S, ||\cdot||, s^*\}$ be an initial epistemic model for n-pin DMM game model G, if $|F_G| = m \cdot n$, then $|S| = m^n$.

Proof. This comes from basic combinatorial calculations. Since G is a n-pin game, $|F_G| = m \cdot n$ means that there are m types of available flowers pegs. A flower sequence, represented by a flower configuration, has n spots, and each of the n spots can choose from m types of flower pegs. Hence, there are m^n combinations. □

Following from this proof, each possible world in an epistemic model of a DMM game item represents an unique flower configuration C. Formally, for any $s, s' \in S$ such that $s \models C, s' \models C'$, if $C = C'$, then $s = s'$. We write C_s to denote the flower configuration C that is true at world s.

Precondition

Epistemic models represent an agent's beliefs about possible flower configurations. Given a game model G, the initial epistemic model contains all possible flower configurations. When an agent deliberates over clues, some possibilities get eliminated, because they are inconsistent with information encoded by the clues. After reasoning about all the clues, only one world should remain, namely, the actual world s^* where *goal* is true.

To model this reasoning process, we define event models $\mathbf{E}_L$ to represent the action of reasoning about a clue $L \in \mathcal{L}_G$. Given a DMM game model G, we define an event model $\mathbf{E}_L = \{e_L, pre\}$ where e_L is the action of observing clue $L \in \mathcal{L}_G$, and pre is the precondition for clue L to be true. For 2-pin DMM game models, we define four preconditions pre with respect to each feedback sequence. Consider any clue $L = p_1 \wedge q_2 \wedge \sigma^2$, then

$$\begin{aligned}
&\text{If } \sigma^2 = oo, && pre_{e_L} = q_1 \wedge p_2\\
&\text{If } \sigma^2 = rr, && pre_{e_L} = \neg p_1 \wedge \neg p_2 \wedge \neg q_1 \wedge \neg q_2\\
&\text{If } \sigma^2 = gr, && pre_{e_L} = (p_1 \wedge \neg q_2) \vee (q_2 \wedge \neg p_1)\\
&\text{If } \sigma^2 = or, && pre_{e_L} = (p_2 \wedge \neg p_1 \wedge \neg q_1) \vee (q_1 \wedge \neg p_2 \wedge \neg q_2).
\end{aligned}$$[7]

The precondition for $\sigma^2 = oo$ says that clue $L = p_1 \wedge q_2 \wedge oo$ is true at worlds where the position of the two flower pegs in L are switched. The precondition for $\sigma^2 = rr$ says that $L = p_1 \wedge q_2 \wedge rr$ is true at worlds where neither of the flower pegs in L appear in this world. The precondition for $\sigma^2 = gr$ says that $L = p_1 \wedge q_2 \wedge gr$ is true at worlds where one of the flower pegs in L is at the right position and another flower peg does not appear. Finally, the precondition for $\sigma^2 = or$ says that clue $L = p_1 \wedge q_2 \wedge or$ is true at worlds where one of the flower pegs in L is the right flower at another position, and the other position holds neither flower pegs in L.

The product model $\mathbf{S} \otimes \mathbf{E}$ is defined as usual in DEL.

Now we need to show that the preconditions indeed capture the correct interpretation of each feedback, which we do by defining a notion of compatibility.

Definition 4.8 (Compatibility)**.** A possible world s is compatible with a clue $L = C \wedge \sigma^n$ if $f^{C_s}(C) = \sigma^n$.

Recall that C_s is the flower configuration C that is true at world s. $f^{C_s}(C) = \sigma^n$ is an extension of the feedback function in Definition 4.7, and is defined by replacing *goal* in Definition 4.7 with C_s. From the definition of the valuation, we know that each possible world s in an epistemic model holds a unique flower configuration C_s. Therefore, $f^{C_s}(C) = \sigma^n$ means that if s is the actual world, and C is the flower configuration in clue $L = C \wedge \sigma^n$, we get a feedback sequence $\sigma^{n'}$ such that $\sigma^{n'} = \sigma^n$.

Proposition 4.2. Let $\mathbf{S}_0$ be an initial epistemic model for a game model G, $\mathbf{E}_L = \{e_L, pre\}$, where e_L is the action of observing a clue L, then for $\mathbf{S}_1 = \mathbf{S}_0 \otimes \mathbf{E}_L$ and $S_1 \in \mathbf{S}_1$, world $s \in S_1$ if and only if s is compatible with L.

[7] According to the feedback function, the precondition for the *or* feedback to flower configuration $p_1 \wedge q_2$ can also be $p_2 \wedge \neg p_1 \wedge \neg q_1 \wedge \neg q_2$, and is equivalent to the precondition defined in the text, because p_1 already indicates the q_2 cannot be the case, according to the condition of being a flower configuration (see Definition 4.5).

Proof. Without loss of generality, assume $L = a_1 \wedge b_2 \wedge \sigma^2$. By the definitions of the DEL models in Section 4.1 and 4.2.2, $S_1 = \{s | s \in S_0 \otimes E\}$ and $S_0 \otimes E = \{(s,e) | s \models pre\}$. Note that (s,e) represents the possible world s with label (s,e), hence, $s \in S_1 \Leftrightarrow s \models pre$. Hence, we need to show that for all $s \in S_1$:

- $\Rightarrow$: if $s \in S_1$, then $f^{C_s}(a_1 \wedge b_2) = \sigma^2$, and
- $\Leftarrow$: if $s \notin S_1$, then $f^{C_s}(a_1 \wedge b_2) \neq \sigma^2$.

Let us check the four feedback sequences oo, rr, gr, or one by one.

(i) $\sigma^2 = oo$:

- $\Rightarrow$: given any $s \in S_1$, by definition of pre, $s \models a_2 \wedge b_1$. By definition of feedback sequence, $f^{C_s}(a_1 \wedge b_2) = oo$.
- $\Leftarrow$: given any $s \notin S_1$, by definition of pre, $s \not\models a_2 \wedge b_1$. Hence, $s \models \neg a_2 \vee \neg b_1$, therefore, $s \models \neg a_2$ or $s \models \neg b_1$. If $s \models \neg a_2$, then $s \not\models a_2$, and therefore $f^{C_s}(a_1) \neq o$, so that $f^{C_s}(a_1 \wedge b_2) \neq oo$. If $s \models \neg b_1$, then $s \not\models b_1$, and $f^{C_s}(b_2) \neq o, f^{C_s}(a_1 \wedge b_2) \neq oo$. In all, if $s \notin S_1$, then $f^{C_s}(a_1 \wedge b_2) \neq oo$.

(ii) $\sigma^2 = rr$:

- $\Rightarrow$: given any $s \in S_1$, by definition of pre, $s \models \neg a_1 \wedge \neg a_2 \wedge \neg b_1 \wedge \neg b_2$. By definition of feedback sequence, $f^{C_s}(a_1 \wedge b_2) = rr$.
- $\Leftarrow$: given any $s \notin S_1$, by definition of pre, $s \models a_1 \vee a_2 \vee b_1 \vee b_2$. That is to say, $s \models a_1$ or $s \models a_2$ or $s \models b_1$ or $s \models b_2$. If $s \models a_1$, then $f^{C_s}(a_1) = g, f^{C_s}(a_1 \wedge b_2) \neq rr$. Similarly for $s \models a_2$, $s \models b_1$ or $s \models b_2$. In all, if $s \notin S$, then $f^{C_s}(a_1 \wedge b_2) \neq rr$.

(iii) $\sigma^2 = gr$:

- $\Rightarrow$: given any $s \in S_1$, by definition of pre, $s \models (a_1 \wedge \neg b_2) \vee (b_2 \wedge \neg a_1)$. Hence, $s \models a_1 \wedge \neg b_2$ or $s \models b_2 \wedge \neg a_1$. If $s \models a_1 \wedge \neg b_2$, then $s \models a_1$ and $s \not\models b_2$. By $s \models a_1$, $f^{C_s}(a_1) = g$. By $s \not\models b_2, f^{C_s}(b_2) = r$. Hence, $f^{C_s}(a_1 \wedge b_2) = gr$. Similarly for $s \models b_2 \wedge \neg a_1$, we have $f^{C_s}(b_2) = g, f^{C_s}(a_1) = r$ and $f^{C_s}(a_1 \wedge b_2) = gr$. In all, $f^{C_s}(a_1 \wedge b_2) = gr$.
- $\Leftarrow$: given any $s \notin S_1$, by definition of pre, $s \not\models (a_1 \wedge \neg b_2) \vee (b_2 \wedge \neg a_1)$, hence, $s \models (\neg a_1 \vee b_2) \wedge (\neg b_2 \vee a_1)$. Suppose that $s \models \neg a_1$, then $s \models \neg b_2$, then $f^{C_s}(a_1) \neq g$ and $f^{C_s}(b_2) \neq g$, therefore $f^{C_s}(a_1 \wedge b_2) \neq gr$. Suppose that $s \models b_2$, then $s \models a_1$. Hence, $f^{C_s}(a_1) = g, f^{C_s}(b_2) = g$, therefore $f^{C_s}(a_1 \wedge b_2) = gg$ and $f^{C_s}(a_1 \wedge b_2) \neq gr$.

(iv) $\sigma^2 = or$:

- $\Rightarrow$: given any $s \in S_1$, by definition of *pre*, $s \models (b_1 \wedge \neg a_2 \wedge \neg b_2) \vee (a_2 \wedge \neg a_1 \wedge \neg b_1)$. Hence, $s \models b_1 \wedge \neg a_2 \wedge \neg b_2$ or $s \models a_2 \wedge \neg a_1 \wedge \neg b_1$. Suppose that $s \models b_1 \wedge \neg a_2 \wedge \neg b_2$, by Definition 4.7, $f^{C_s}(a_1) = r$ and $f^{C_s}(b_2) = o$. Hence, $f^{C_s}(a_1 \wedge b_2) = or$. Similarly for $s \models a_2 \wedge \neg a_1 \wedge \neg b_1$. In all, $f^{C_s}(a_1 \wedge b_2) = or$.

- $\Leftarrow$: given any $s \notin S_1$, by definition of *pre*, $s \models (\neg b_1 \vee a_2 \vee b_2) \wedge (\neg a_2 \vee a_1 \vee b_1)$. Hence, $s \models \neg b_1 \vee a_2 \vee b_2$ and $s \models \neg a_2 \vee a_1 \vee b_1$. Suppose that $s \models b_1$, then $s \models a_2 \vee b_2$. If $s \models a_2$, then $C_s = b_1 \wedge a_2$ and $f^{C_s}(a_1 \wedge b_2) = oo \neq or$. If $s \models b_2$, then $C_s = b_1 \wedge b_2$ and $f^{C_s}(a_1 \wedge b_2) = gr \neq or$. Hence, if $s \models b_1$ then $f^{C_s}(a_1 \wedge b_2) \neq or$.

  Suppose that $s \models \neg b_1$, then $s \models \neg a_2 \vee a_1$. If $a \models \neg a_2$, then $s \models \neg b_1 \wedge \neg a_2$, and $f^{C_s}(a_1 \wedge b_2) \neq or$ because there cannot be any o feedback for either flower. If $s \models a_1$, then $f^{C_s}(a_1) = g$ and $f^{C_s}(a_1 \wedge b_2) \neq or$. In all, $f^{C_s}(a_1 \wedge b_2) \neq or$.

Therefore, in all, for any $L = C \wedge \sigma^2$ and $\mathbf{S}_1 = \mathbf{S}_0 \otimes \mathbf{E}$,

$$s \in S_1 \Leftrightarrow f^{C_s}(C) = \sigma^2.$$

□

Because of Proposition 4.2 and uniqueness of *goal* in the game setting, it follows that after updating with all clues, the DEL model converges to *goal*.

Theorem 4.1. For any 2-pin DMM game model $G = (F_G, \mathcal{L}_G, goal)$ where $\mathcal{L}_G = \{L_1, \ldots, L_n\}$, let $\mathbf{E}_i = \{e_i, pre\}$ where action e_i is observing clue $L_i \in \mathcal{L}_G$, then for $\mathbf{S}_n = \mathbf{S}_0 \otimes \mathbf{E}_1 \otimes \ldots \mathbf{E}_n$, $|S_n| = 1$.

Proof. By Proposition 4.2, it is always the case that the actual world $s^* \in S_n$. Hence, $|S_n| \geq 1$. By uniqueness of *goal*, for all $s \in |S_n|$ such that s is compatible with all clues in $\mathcal{L}_G$, $s = s^*$. Hence, $|S_n| = 1$. □

4.3 Sequential Update

The DEL model of a DMM game model generates a sequence of epistemic models, and each one contains some possible flower configurations that are compatible with the information processed so far. Agents may reason about clues one by one, or reason about several clues at the same time. In this section, we consider the first case, and in the next section the second case.

4.3.1 Illustration

Reasoning about clues one by one generates a sequence of epistemic models that shrink in size until only one possible world remains.

Example 4.5. Consider the game model G in Example 4.3, where the initial epistemic model is $\mathbf{S}_0 = \{S_0, ||\cdot||, s^*\}$, and $|S_0| = 4^2 = 16$. There are 16 possible flower configurations, each true at a unique possible world $s_i \in S_0$:

$$\begin{array}{llll} s_1 \models a_1 \wedge a_2, & s_2 \models a_1 \wedge b_2, & s_3 \models a_1 \wedge c_2, & s_4 \models a_1 \wedge d_2, \\ s_5 \models b_1 \wedge a_2, & s_6 \models b_1 \wedge b_2, & s_7 \models b_1 \wedge c_2, & s_8 \models b_1 \wedge d_2, \\ s_9 \models c_1 \wedge a_2, & s_{10} \models c_1 \wedge b_2, & s_{11} \models c_1 \wedge c_2, & s_{12} \models c_1 \wedge d_2, \\ s_{13} \models d_1 \wedge a_2, & s_{14} \models d_1 \wedge b_2, & s_{15} \models d_1 \wedge c_2, & s_{16} \models d_1 \wedge d_2. \end{array}$$

Let the following event models encode the two clues in the game model:
$\mathbf{E}_{L_1} = \{e_{L_1}, pre_{L_1}\}$, since $L_1 = b_1 \wedge b_2 \wedge gr$, $pre_{L_1} = (b_1 \wedge \neg b_2) \vee (b_2 \wedge \neg b_1)$, and
$\mathbf{E}_{L_2} = \{e_{L_2}, pre_{L_2}\}$, since $L_2 = b_1 \wedge d_2 \wedge oo$, $pre_{L_2} = d_1 \wedge b_2$.

If the agent first reasons about clue L_1, then the updated epistemic model is $\mathbf{S}_1 = \mathbf{S}_0 \otimes \mathbf{E}_{L_1}, S_1 = \{s_2, s_5, s_7, s_8, s_{10}, s_{14}\}$.

And reasoning about clue L_2 results in the second updated epistemic model $\mathbf{S}_2 = \mathbf{S}_1 \otimes \mathbf{E}_{L_2}, S_2 = \{s_{14}\}$.

Since $|S_2| = 1$, and $s^* = s_{14}$, the *goal* is $d_1 \wedge b_2$.

In Example 4.5, the initial epistemic model $\mathbf{S}_0$ has 16 possible worlds, and after updating with clue L_1, $\mathbf{S}_1$ has 6 possible worlds. Updating with clue L_2 results in $\mathbf{S}_2$ where only the actually world is left. S_0, S_1 and S_2 form a sequence of updated epistemic models, which we illustrate in Figure 4.1.

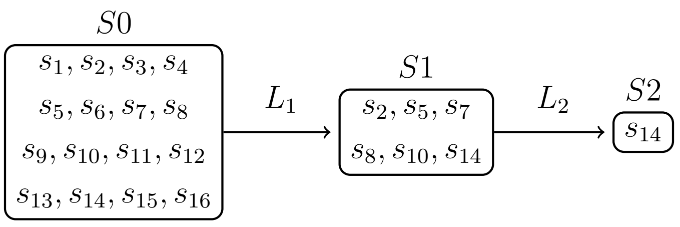


Figure 4.1: Illustration of the update sequence for Example 4.5

We call such a sequence of updated epistemic models an "update sequence". Formally,

Definition 4.9 (Update sequence)**.** Let G be a (2-pin) DMM game model, $\mathbf{S}_0$ be the initial epistemic model, and $\mathbf{E}_1, \ldots, \mathbf{E}_n$ be event models for clues $L_1, \ldots, L_n \in \mathcal{L}$, then $\mathbf{S}_1 = \mathbf{S}_0 \otimes \mathbf{E}_1, \mathbf{S}_2 = \mathbf{S}_1 \otimes \mathbf{E}_2, \ldots, \mathbf{S}_n = \mathbf{S}_{n-1} \otimes \mathbf{E}_n$. An update sequence of this game model is $\mathcal{Q}_G = \langle S_0, S_1, \ldots, S_n \rangle$ where $S_k \in \mathbf{S}_k$.

As stated in Theorem 4.1, the update sequence of any DMM game model will end with the epistemic model of size 1, and for any epistemic models S, S' in the same update sequence, if S is before S', then $|S| \geq |S'|$.

4.3.2 Complexity Measurements

We now define several criteria to measure the complexity of an update sequence generated by the DEL model of a DMM game model. We measure the complexity of an update sequence from the perspective of the number of possible worlds and the rate of convergence of the epistemic models respectively.

Size of epistemic models

One natural proposal is to take the sum of the sizes of all epistemic models in an update sequence. The size of an epistemic model reflects how many possibilities could serve as the correct answer, and hypothetically the more possibilities, the more difficult of this game item. Therefore, if a game item generates an update sequence in which each epistemic model contains many possible worlds, then this game is perceived to be more difficult. Formally,

Definition 4.10 (Measurement SUM_0)**.** The SUM_0 complexity of an update sequence $\mathcal{Q} = \langle S_0, \dots, S_n \rangle$ is defined as

$$\mathrm{SUM}_0 := \sum_{i=0}^{n} |S_i|$$

We illustrate this measurement with the following example.

Example 4.6. Consider the following two game items.

(1) **Game item 125966** (Rating: -15.77):

$G_1 = \{F_1, \mathcal{L}_1, goal_1\}$, $F_1 = \{a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2, e_1, e_2\}, |F_1| = 5 \cdot 2 = 10$,

$\mathcal{L}_1 = \{L_1^1, L_2^1\}$, $L_1^1 = a_1 \wedge b_2 \wedge rr$, and $L_1^2 = c_1 \wedge d_2 \wedge rr$.

(2) **Game item 125780** (Rating: -30.15):

$G_2 = \{F_2, \mathcal{L}_2, goal_2\}$, $F_2 = \{a_1, a_2, b_1, b_2, c_1, c_2\}$, $|F_2| = 3 \cdot 2 = 6$,

$\mathcal{L}_2 = \{L_1^2, L_2^2\}$, $L_1^2 = a_1 \wedge a_2 \wedge gr$, and $L_2^2 = b_1 \wedge b_2 \wedge rr$.

The sizes of the epistemic models in the update sequence for G_1 are $\langle 25, 9, 1 \rangle$, and for G_2 they are $\langle 9, 4, 1 \rangle$. According to Definition 4.10, $\mathrm{SUM}_0(G_1) = 25 + 9 + 1 = 35$ and $\mathrm{SUM}_0(G_2) = 9 + 4 + 1 = 14$.

The Elo rating for G_1 is -15.77 and the rating for G_2 is -30.15, which means that empirically G_1 is harder than G_2. The SUM_0 measurement for these two game items reflects this fact, with G_1 of complexity measurement 35 and G_2 of 14.

However, measurement SUM_0 may overate the influence of the initial model, which we demonstrate in the next example.

Example 4.7. Consider game items (3) and (4), which have different sizes of initial epistemic models and therefore very different SUM_0 measurements. However, their ratings are very similar to each other.

(3) **Game item 125831** (Rating: -20.84):

$G_3 = \{F_3, \mathcal{L}_3, goal_3\}$, $F_3 = \{a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2\}, |F_3| = 4 \cdot 2 = 8,$

$\mathcal{L}_3 = \{L^3_1, L^3_2\}$, $L^3_1 = a_1 \wedge b_2 \wedge rr$, $L^3_1 = c_1 \wedge d_2 \wedge oo.$

(4) **Game item 125563** (Rating: -22.66):

$G_4 = \{F_4, \mathcal{L}_4, goal_4\}$, $F_4 = \{a_1, a_2, b_1, b_2, c_1, c_2\}$, $|F_4| = 3 \cdot 2 = 6,$

$\mathcal{L}_4 = \{L^4_1, L^4_2\}$, $L^4_1 = a_1 \wedge a_2 \wedge gr$, $L^4_1 = b_1 \wedge c_2 \wedge rr.$

The sizes of epistemic models in DEL structure for G_3 is $\langle 16, 4, 1 \rangle$, and the sizes of epistemic models in the DEL structure for G_4 is $\langle 9, 4, 1 \rangle$.

According to Definition 4.10, $\text{SUM}_0(G_3) = 16+4+1 = 21$ and $\text{SUM}_0(G_4) = 9+4+1 = 14$. However, the Elo rating of these two game items are very close to each other (-20.84 and -22.66). We therefore propose another measurement SUM_1 that excludes the size of the initial epistemic models because the initial epistemic model simply contains all possibilities, which does not directly relate to how to solve the game. Formally,

Definition 4.11 (SUM_1)**.** The SUM_1 complexity of an update sequence $\mathcal{Q} = \langle S_0, \ldots, S_n \rangle$ is defined as

$$\text{SUM}_1 := \sum_{i=1}^{n} |S_i|$$

This SUM_1 measurement counts the number of possible worlds in an update sequence without counting the initial epistemic state, which avoids the situation illustrated in Example 4.7. According to this new measurement, the SUM_1 measurement for the update sequence of game model G_3 is 5, which equals the SUM_1 measurement for DEL structure of game model G_4. This fits better with the Elo ratings for both items. We test whether the SUM_1 measurement captures the empirical difficulty better than the SUM_0 measurement in the next section.

In addition to the two sum measurements, we also define the following measurement that considers the average size of epistemic models in an update sequence.

Definition 4.12 (SV)**.** The SV complexity of an update sequence $\mathcal{Q} = \langle S_0, \ldots, S_n \rangle$ is defined as

$$\text{SV} := \frac{\text{SUM}_0(\mathcal{Q})}{n}$$

Here, "V" stands for "average", indicating that the SV measurement considers the average number of possible worlds per update. The higher the SV measurement, the more difficult a game item is.

Convergence rate

Another measurement of the complexity of DEL update sequences is the rate of convergence. If from $\mathbf{S}_{k-1}$ to $\mathbf{S}_k$, a clue L_k eliminates more states, then the faster this game item converges to *goal*. Our assumption is the more possible worlds a clue eliminates, the more difficult it is for a player to reason about this clue.

Definition 4.13 (CR)**.** The CR complexity of an update sequence $\mathcal{Q} = \langle S_0, \ldots, S_n \rangle$ is defined as

$$\text{CR} := \sum_{i=1}^{n} \frac{|S_{i-1}|}{|S_i|}$$

The CR measurement takes the sum of the convergence rate of an update sequence. The more states that get eliminated, the more reasoning is required for this step, and therefore we expect a higher rating. When increasing the number of states S_i eliminates from S_{i-1}, $\frac{|S_{i-1}|}{|S_i|}$ increases, resulting in a larger value that indicates a higher cognitive difficulty of this game item.

Consider the four game models in Example 4.6 and Example 4.7, the CR complexity for each DEL structure of those game models are as follows:

$$\begin{aligned}
\text{CR}(\mathcal{Q}_1) &= \frac{25}{9} + \frac{9}{1} \approx 11.78, \\
\text{CR}(\mathcal{Q}_2) &= \frac{9}{4} + \frac{4}{1} \approx 6.25, \\
\text{CR}(\mathcal{Q}_3) &= \frac{16}{4} + \frac{4}{1} \approx 8, \text{ and} \\
\text{CR}(\mathcal{Q}_4) &= \frac{9}{4} + \frac{4}{1} \approx 6.25.
\end{aligned}$$

The CR measurement says that game model G_1 is the most complex, and then G_3 the next complex, and that game models G_2 and G_4 are equally complex. As for the Elo ratings, the difficulty ranking of the four game models is $G_1 > G_4 > G_3 > G_2$. CR measurement captures the fact the G_1 is more difficult than the rest, and that G_3 is more difficult than G_2, but not G_4 is more difficult than G_2 and G_3, and that G_2 is the least difficult.

4.4 Intersecting Update

As shown earlier, an update sequence is order-dependent. That is to say, the order of which clue to process decides the update sequence, and consequently influences the value of the complexity measurements of that game item. However, experiments that track eye movements reveal that people do not always follow a specific order in reasoning over clues (Truescu, 2016). Therefore, since assuming the order of clues to update in a DEL update is too strong, and we want to relax or drop this constraint on our DEL model.

Fortunately, updating clues one after one is not the only way to solve a DMM game item in the language of DEL. In fact, an agent can update the initial epistemic model using all the clues at the same time, and then take the intersection of such updates. According to the DEL formalization, this procedure always leads to *goal* as well.

Proposition 4.3. For a 2-pin DMM game model G, let $\mathbf{S}$ be an epistemic model, $\mathbf{E}_L$ be an event model for clue $L \in \mathcal{L}_G$, and $\mathbf{E}_{L'}$ an event model for clue $L' \in \mathcal{L}_G$, then:

$$\mathbf{S} \otimes \mathbf{E}_L \otimes \mathbf{E}_{L'} = \mathbf{S} \otimes \mathbf{E}_{L'} \otimes \mathbf{E}_L.$$

Proof. Let $A = \mathbf{S} \otimes \mathbf{E}_L \otimes \mathbf{E}_{L'}$, and $B = \mathbf{S} \otimes \mathbf{E}_{L'} \otimes \mathbf{E}_L$. Let $\varphi = \bigvee C^s$ for all $s \in S$, then for all $s \in S$, $s \models \varphi$. By definition of product update, for all $a \in A, a \models \varphi \wedge pre_{e_L} \wedge pre_{e_{L'}}$, and for all $b \in B, b \models \varphi \wedge pre_{e_{L'}} \wedge pre_{e_L}$. Since all the preconditions *pre* do not contain any modal operator, but are just propositional logic formulas, whether *pre* is true in some world s depends only on the valuation at s, and not at any other world. By propositional logic, $A = B$. □

Proposition 4.3 shows that the order of an update does not effect the updated model. This proposition can be written in a different way:

Theorem 4.2. For a 2-pin DMM game model G, let $\mathbf{S}_0$ be the initial epistemic model and $\mathbf{E}_{L_1}, \ldots \mathbf{E}_{L_n}$ be event models for clue $L, \ldots, L_n$:

$$\mathbf{S}_0 \otimes \mathbf{E}_{L_1} \ldots \otimes \mathbf{E}_{L_n} = \bigcap_{k=1}^{n} \mathbf{S}_0 \otimes \mathbf{E}_{L_k}$$

Theorem 4.2 says that to update epistemic models clue after clue is the same as updating all clues at the same time. In particular, let the initial epistemic model be updated with each clue, and simply take the intersection of such update to get *goal*. The proof for Theorem 4.2 is similar to the one for Proposition 4.3, which uses the fact that in propositional logic if $p = a \wedge b$ and $q = c \wedge d$, then $p \wedge q = a \wedge b \wedge c \wedge d$.

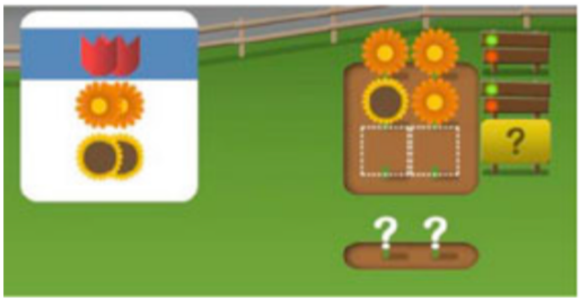


Figure 4.2: Screenshot of a 2-pin DMM game item

4.4.1 Illustration

Now we show by way of an example that we can find *goal* for a game item by updating each clue with the initial epistemic model and then taking the intersection of them.

Example 4.8. Consider the game item in Figure 4.2, which has game model $G = \langle F, \mathcal{L}, goal\rangle$, where $F = \{a_1, a_2, b_1, b_2, c_1, c_2\}$, $\mathcal{L} = \{L_1 L_2\}$, $L_1 = b_1 \wedge b_2 \wedge gr$, and $L_2 = c_1 \wedge b_2 \wedge gr$. The initial epistemic model $\mathbf{S}_0 = \{S_0, ||\cdot||, s^*\}, |S_0| = 3^2 = 9$. There are 9 possible flower configurations, each true at a possible world $s_i \in S_0$:

$$\begin{array}{lll} s_1 \models a_1 \wedge a_2, & s_2 \models a_1 \wedge b_2, & s_3 \models a_1 \wedge c_2, \\ s_4 \models b_1 \wedge a_2, & s_5 \models b_1 \wedge b_2, & s_6 \models b_1 \wedge c_2, \\ s_7 \models c_1 \wedge a_2, & s_8 \models c_1 \wedge b_2, & s_9 \models c_1 \wedge c_2. \end{array}$$

Let event models encode the two clues in the game model as follows:
$\mathbf{E}_{L_1} = \{e_{L_1}, pre_{L_1}\}$, since $L_1 = b_1 \wedge b_2 \wedge gr$, $pre_{L_1} = (b_1 \wedge \neg b_2) \vee (b_2 \wedge \neg b_1)$.
$\mathbf{E}_{L_2} = \{e_{L_2}, pre_{L_2}\}$, since $L_2 = c_1 \wedge b_2 \wedge gr$, $pre_{L_2} = (c_1 \wedge \neg b_2) \vee (b_2 \wedge \neg c_1)$.
Updating the initial epistemic model with clue L_1 results in the updated epistemic model $\mathbf{S}_1 = \mathbf{S}_0 \otimes \mathbf{E}_{L_1}, S_1 = \{s_2, s_4, s_6, s_8\}$.

And updating the initial epistemic model with clue L_2 results in the updated epistemic model $\mathbf{S}_2 = \mathbf{S}_0 \otimes \mathbf{E}_{L_2}, S_2 = \{s_2, s_5, s_7, s_9\}$.

Intersecting $\mathbf{S}_1$ and $\mathbf{S}_2$ leads to the actual world, $S_1 \cap S_2 = \{s_2\}, s^* = s_2, goal = a_1 \wedge b_2$.

Figure 4.3 shows the update sequence for Example 4.8. In this case, the initial epistemic model is updated with each clue individually, and then the intersection of all the updated models is taken, which leads to the secret sequence *goal*.

This leads to the notion of an *unordered update sequence*, which contrasts with the notion of update sequence where clues are processed one after one. The unordered update sequence assumes that it is equally likely to process any clue at first, and *goal* is derived by taking the intersection of such updates. Formally,

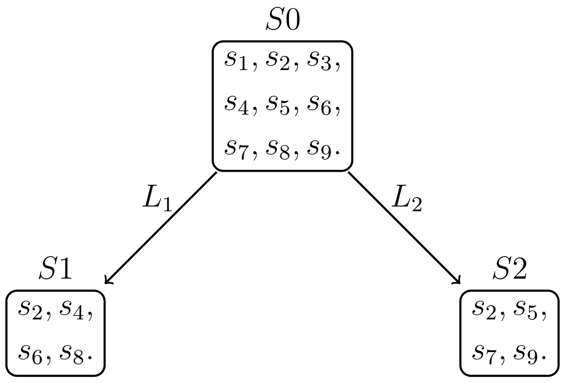


Figure 4.3: Illustration of the update sequence for Example 4.8

Definition 4.14 (Unordered update sequence)**.** For any event models $\mathbf{E}_i \in \{\mathbf{E}_1, \dots, \mathbf{E}_n\}$, an unordered update sequence is a set $\mathcal{U} = \langle S_0, \dots, S_n \rangle$ where $S_n \in \mathbf{S}_n$ such that $\mathbf{S}_n = \mathbf{S}_0 \otimes \mathbf{E}_n$.

4.4.2 Complexity Measurements

There are several ways of measuring the complexity of an unordered update sequence. On the one hand, the measurements we defined for update sequence can be applied to unordered update sequence with minor modifications. On the other hand, we can also measure complexity of an unordered update sequence with respect to different feedback types.

Complexity measurements SUM$_0$, SUM$_1$ and SV can be applied on unordered update sequences. The only change is that whereas S_i in an ordered update sequence refers to the i-the iteration, in an unordered update sequence it simply refers to an update of the initial epistemic model, but the calculations are the same. The complexity measurement CR is modified as follows:

$$\text{CR'} := \sum_{i=1}^{n} \frac{|S_i|}{|S_0|},$$

because for each $S_i, i > 0$ is the result of updating clue L_i against the initial epistemic model, hence we should take the ratio of $|S_i|$ against $|S_0|$ accordingly.

To measure complexity of an unordered update sequence with respect to different feedback types, the idea is straightforward. We measure how many possible worlds a clue keeps after updating, and average this number by feedback types. For example, the sizes of epistemic models in the unordered update sequence in Example 4.8 are 9, 4, and 4, and both clues contain gr feedbacks. For each clue, the number of possible worlds kept by the clue is 4. Since both clues contain gr, the complexity measurement for gr feedback is $(4+4)/2 = 4$, and the complexity measurements for the other three feedback types are all

0. Besides computing the number of possible worlds that survive an elimination, we can also compute the ratio of the number of worlds that survive to the number of worlds in the initial epistemic model, as an echo to the concern about convergence rate discussed in earlier sections. Listing 4.1 shows the `R` code of the algorithm that we use in computing these measurements for the unordered update sequence for each 2-pin DMM game item.

Listing 4.1: `R` code for measurements over unordered update sequence

```
for (i in 1 : nrow(2_pin_DMM)){
        if (clue[i] == oo){
                oo <- oo + x
        } else if (clue[i] == rr){
                rr <- rr + x
        } else if (clue[i] == gr){
                gr <- gr + x
        } else if (clue[i] == or){
                or <- or + x
        }
}
```

We define two complexity measurements using the this algorithm. One is the DELs measurement, which considers the number of possible worlds that get selected per feedback type, and the other is the DELr measurement, which considers the ratio of the selected possible worlds to the initial epistemic model per feedback type. We first compute a value `x` for each clue, and then average `x` with respect to feedback types. For measurement DELs, `x` is the size of epistemic model $\mathbf{S}_i$ after updating the initial epistemic model with event model $\mathbf{E}_i$ that contains clue L_i. For measurement DELr, `x` is calculated as `|S_i| / |S_0|` for a given DEL structure. All feedbacks `oo,rr,gr,or` are initialized as `0`. Hence, if a feedback σ does not appear in a game model, the evaluation for feedback σ remains 0. Both the DELs and DELr measurements are averaged by the number of appearance of that feedback.

Consider the game item in Example 4.5. The size of the initial epistemic model is 16. One clue has a *gr* feedback, which leads to an updated epistemic model of 6 possible worlds. Another clue has a *oo* feedback, which leads to an updated epistemic model of only 1 possible world. The unordered update sequence for this game item is $\langle 16, 6, 1 \rangle$, with 6 for *gr* and 1 for *oo*. Hence, the DELs measurement for this item is

$$\langle oo = 1, rr = 0, gr = 6, or = 0 \rangle$$

And the DELr measurement for this item is

$$\langle oo = \frac{1}{16}, rr = 0, gr = \frac{6}{16}, or = 0 \rangle$$

Table 4.1 lists the six complexity measurements we have defined so far. The first four measurements apply to both update sequences and unordered update sequences, and the last two measurements only apply to unordered update sequences and take feedback type into account. In the next chapter, we test these measurements with the empirical dataset of item ratings.

Table 4.1: Summary of complexity measurements

Complexity Measurement	SUM_0	SUM_1	SV	CR	DELs	DELr
Definition	$\sum_{i=0}^{n} \lvert S_i \rvert$	$\sum_{i=i}^{n} \lvert S_i \rvert$	$\frac{\mathrm{SUM}_0}{n}$	$\sum_{i=0}^{n} \frac{\lvert S_{i-1} \rvert}{\lvert S_i \rvert}$	SUM_0 per feedback type	CR per feedback type

4.5 Informativeness of feedbacks

Before testing our complexity measurements with empirical dataset, let us pause a bit in and take a closer look at how each clue eliminates possible worlds differently in the elimination process.

The number of possible worlds eliminated relates to the notion of informativeness. In information theory, informativeness of the occurrence of an event is defined by its entropy. The less likely the occurrence of the event is, the more information it contains. Entropy can also be viewed as the bits needed to encode the occurrence of the information. It was first studied by Shannon (2001), and then developed into an important area of study. Measuring informativeness of information based on the probability distribution is used in many research areas. In particular, Fangzhou et al. (2015) used this measurement to simulate human reasoning of logic rules in syllogistic reasoning. In the DEL models of 2-pin DMM game items, a natural correspondence to informativeness of clues is the number of possible worlds eliminated by an update. Assume that an agent is currently at epistemic model $\mathbf{S}$ with $|S| = m$. From Section 4.2, for each $s \in S, s \models C^s$ where C^s is a possible flower configuration that could be an answer of this game. Hence, the probability of each s being s^*

$$P(s = s^*) := \frac{1}{|S|} \tag{4.1}$$

When $|S| = 1, P(s^* = s^*) = 1$, in accordance with Theorem 4.1.

Definition 4.15 (Elimination power of clues)**.** Let $\mathbf{E}$ be an event model where e_L is the action of observing clue L, $\mathbf{S}_0$ is the initial epistemic model before updating with $\mathbf{E}$, and $\mathbf{S} = \mathbf{S}_0 \otimes \mathbf{E}$, then the elimination power of clue L is defined as

$$El(L) := \left(\frac{|S|}{|S_0|}\right)^{-1}$$

by Proposition 4.1, $|S_0| > 0$.

Similar to the least tree approach in the tableaux method, the elimination power of clues can provide a ranking of difficulty of game items. The stronger the elimination power a clue has, the faster it helps a model to converge to the actual world.

Claim 4.1. The elimination power of a clue L in game model G is dependent on the feedback types σ in clue L and the number of flower types $|F| = m \cdot n$ in the game model G.

Proof. By Definition 4.6 and Proposition 4.1, for any 2-pin game model G, $|F| = 2 \cdot m$. For any clue $L = a_1 \wedge b_2 \wedge \sigma$ in G, there are four cases to consider.

(i) $\sigma = oo$:

Recall that $pre_{e_L} = b_1 \wedge a_2$. Hence, in $\mathbf{S}_1, S_1 = \{(b_1, a_2)\}$, therefore, $|S_1| = 1$, and $El(L) = m^2$.

(ii) $\sigma = rr$:

The precondition for $\mathbf{E}$ is $pre_{e_L} = \neg a_1 \wedge \neg a_2 \wedge \neg b_1 \wedge \neg b_2$. Hence,

$$|S_1| = \begin{cases} (m-1)^2 & \text{if } a = b \\ (m-2)^2 & \text{if } a \neq b \end{cases}$$

This comes straightforward from basic combinatorial operations. For $a = b$, only $m-1$ types of flowers are still possible in making possible flower configurations, and for $a \neq b$, there are $m-2$ types of flowers left. By similar reasoning as in Proposition 4.1, we derive the above equations. Accordingly,

$$El(L) = \begin{cases} (\frac{m}{m-1})^2 & \text{if } a = b \\ (\frac{m}{m-2})^2 & \text{if } a \neq b \end{cases}$$

(iii) $\sigma = gr$:

The precondition for this clue says that $pre_{e_L} = (a_1 \wedge \neg b_2) \vee (b_2 \wedge \neg a_1)$. There are two cases to consider, either the first flower is the right flower in the right position and the other one is a wrong flower, or the other way around. For each case, there are $|m-1|$ possible flower configurations, and altogether $|S_1| = 2(m-1)$. Hence,

$$El(L) = \frac{m^2}{2(m-1)}$$

(iv) $\sigma = or$:

The precondition for clues with *or* feedbacks is $pre_{e_L} = (a_2 \wedge \neg a_1 \wedge \neg b_1) \vee (b_1 \wedge \neg b_2 \wedge \neg a_2)$. Similarly to the *gr* case, there are two cases to consider. For each case, there are $m - 2$ possible flower configurations left in the updated model. Hence, in total $|S_1| = 2(m-2)$, and

$$El(L) = \frac{m^2}{2(m-2)}$$

$\square$

We have now proven that the elimination power of a clue L in game model G depends on the feedback types σ in clue L, and the number of flower types $|F|$ in the game model G. For each $m \in [2, 5]$ according to the game design, the elimination power is listed in Table 4.2.

Table 4.2: Elimination power of clues

	oo	rr		gr	or
		$a = b$	$a \neq b$		
$m = 2$	4/1	4/2	-	4/2	-
$m = 3$	9/1	9/4	9/1	9/4	9/2
$m = 4$	16/1	16/9	16/4	16/6	16/4
$m = 5$	25/1	25/16	25/9	25/8	25/6

Similar to the difficulty order of feedbacks in the tableaux model, there is a refined difficulty order of elimination power of feedbacks according to the DEL model.

- when $m = 2$: $|oo| > |rr| = |gr|$.
- when $m = 3$:
  - if the two flowers in the rr clue are the same: $|oo| > |or| > |rr| = |gr|$.
  - if the two flowers in rr clue are different: $|oo| = |rr| > |or| > |gr|$.
- when $m = 4$:
  - if the two flowers in the rr clue are the same: $|oo| > |or| > |gr| > |rr|$.
  - if the two flowers in rr clue are different: $|oo| > |rr| = |or| > |gr|$.
- when $m = 5$: $|oo| > |or| > |gr| > |rr|$.

In all, the informativeness of a clue in the DEL model depends on the number of available flower types and content of the clue, and it provides a finer ranking of feedback types than the tree-difficulty of feedback types in the tableaux model.

4.6 Generality

In this section, we present an algorithm that translates tableaux decision trees of DMM game items to DEL models that follow the sequential update procedure of the same game item. More precisely, we show that the Boolean translation of a game item in the tableaux model is equivalent to the preconditions in its DEL model.

In the tableaux model, each game item is firstly translated into a set of Boolean formulae. Each formula in this set corresponds to a clue in the game. We will show that such a formula in the tableaux model can be treated as an event model in the DEL formalization of DMM. Besides, possible valuations in the tableaux model correspond to possible worlds in the epistemic models in the DEL formalization. Note that the DEL model is able to represent all possible flower configurations, but that is not the case in the tableaux model. If a flower is not used in the clues, the tableaux decision tree will not take that flower into consideration for possible valuations until the last step of searching.[8] We can fix this by forcing a set of all possible valuations for the tableaux model, and this will help with translating a tableaux decision tree to a DEL model.

Boolean formulae and preconditions Firstly, let's have a look at how to translate a Boolean formula $Bt(h, f)$ in the tableaux model into a precondition pre in the DEL model.

1. Map each $c_1, \dots, c_n$ in a tableaux model to an unindexed propositional letter $a, b, \dots$ in the DEL language.

2. For literal $goal(i) = c_n$ in the tableaux model, add index $\cdot_i$ to the propositional letter for c_n in the DEL language. For example $goal(1) = c_1$ in a Boolean formula in the tableaux model can be translated as a_1 in the DEL model.

   As for negation, "$\neq$" in the tableaux model is translated as "$\neg$" in the DEL model. For example $goal(2) \neq c_j$ in a Boolean formula can be translated as $\neg b_2$ in the DEL model.

With these two simple steps, a Boolean formula in the tableaux model can be translated into a precondition for the corresponding clue in the DEL model. Adding a possible world to represent the action of reasoning about that clue, we can derive an event model for this Boolean formula. On the other hand, we can also translate the precondition in

[8]We have mentioned this point earlier, and it will become clearer in the following illustrative examples. Intuitively, one can think of the game item in Figure 3.4. The sunflower is not used in any of the clues, so that it is not represented in the tableaux decision tree of this game item. After searching over the decision tree and finding a consistent valuation, players need to fill in this missing sunflower in order to get the correct answer.

an event model into a Boolean formula following the reverse of this algorithm. In all, the Boolean translation of a game item is equivalent to the event models of this game item.

For example, consider the tableaux formalization in Example 3.1. This game item has two clues, and therefore two Boolean formulae in the tableaux model:

$$Bt(h_1, f_1) = (goal(1) = c_2 \land goal(2) \neq c_2) \lor (goal(2) = c_2 \land goal(1) \neq c_2)$$

and

$$Bt(h_2, f_2) = (goal(1) \neq c_2 \land goal(2) \neq c_4 \land goal(1) = c_4 \land goal(2) = c_2)$$

Following our algorithm, first convert c_2 to propositional letter b, convert c_4 to propositional letter d. Next, we can translate the Boolean formulae as follows.

$$Bt(h_1, f_1) \mapsto (b_1 \land \neg b_2) \lor (b_2 \land \neg b_1)$$

and

$$Bt(h_2, f_2) \mapsto d_1 \land b_2$$

These two formulae correspond to two preconditions in the DEL model. Moreover, every Boolean formula in the tableaux model corresponds to a precondition in the DEL model:

- Feedback *oo*: $goal(1) \neq c_i \land goal(2) \neq c_j \land goal(1) = c_j \land goal(2) = c_i$ in the tableaux model can be translated into $\neg a_1 \land \neg b_2 \land b_1 \land a_2$, and it is equivalent to precondition $a_2 \land b_1$ in the DEL model.

- Feedback *rr*: $goal(1) \neq c_i \land goal(2) \neq c_j \land goal(1) \neq c_j \land goal(2) \neq c_i$ in the tableaux model corresponds to precondition $\neg a_1 \land \neg a_2 \land \neg b_1 \land \neg b_2$ in the DEL model.

- Feedback *gr*: $(goal(1) = c_i \land goal(2) \neq c_j) \lor (goal(2) = c_j \land goal(1) \neq c_i)$ in the tableaux model corresponds to precondition $(a_1 \land \neg b_2) \lor (b_2 \land \neg a_1)$ in the DEL model.

- Feedback *or*: $(goal(1) \neq c_i \land goal(2) \neq c_j) \land ((goal(1) = c_j \land goal(2) \neq c_i) \lor (goal(2) = c_i \land goal(1) \neq c_j))$ in the tableaux model can be translated into $(\neg a_1 \land \neg b_2) \land [(a_2 \land \neg b_1) \lor (b_1 \land \neg a_2)]$, and by propositional logic this is equivalent to $(a_2 \land \neg b_1 \land \neg a_1 \land \neg b_2) \lor (b_1 \land \neg a_2 \land \neg a_1 \land \neg b_2)$, and by definition of flower configurations in the DEL model, this is equivalent to the precondition $(a_2 \land \neg b_1 \land \neg a_1) \lor (b_1 \land \neg a_2 \land \neg b_2)$.

Valuations and possible worlds As mentioned earlier, a valuation in the tableaux model corresponds to a possible world in the epistemic models in the DEL formalization, but there are some possible worlds in the DEL model that do not have direct corresponding valuations in the tableaux model.

$$\begin{tikzcd}[column sep=small]
& & & Bt(h_1, f_1) \arrow[dll, "gr"'] \arrow[drr, "gr"] & & & \\
& \begin{array}{c} goal(1) = c_2 \\ goal(2) \neq c_2 \\ Bt(h_2, f_2) \end{array} \arrow[dl, "gr"'] \arrow[dr, "gr"] & & & & \begin{array}{c} goal(1) \neq c_2 \\ goal(2) = c_2 \\ Bt(h_2, f_2) \end{array} \arrow[dl, "gr"'] \arrow[dr, "gr"] & \\
\begin{array}{c} goal(1) = c_3 \\ goal(2) \neq c_2 \\ \bot \end{array} & & \begin{array}{c} goal(1) \neq c_3 \\ goal(2) = c_2 \\ \bot \end{array} & & \begin{array}{c} goal(1) = c_3 \\ goal(2) \neq c_2 \\ \bot \end{array} & & \begin{array}{c} goal(1) \neq c_3 \\ goal(2) = c_2 \\ \top \end{array}
\end{tikzcd}$$

Figure 4.4: Tableaux tree for the game item in Figure 4.7a

Consider the DMM game item in Figure 4.2. Let $h_1(1) := c_2, h_1(2) := c_2, f_1 = gr$, and $h_2(1) := c_3, h_2(2) := c_2, f_2 = gr$, the tableaux decision tree for this game item is depicted in Figure 4.4.

Figure 4.4 shows that rightmost branch provides a true valuation $goal(1) \neq c_2 \wedge goal(1) \neq c_3 \wedge goal(2) = c_2$. However, this valuation is not enough to tell the secret code. One needs to go back, check the game item, and find $goal(1) = c_1$ to complete this valuation.

If we translate each Boolean formula into preconditions, we can get a sequence of event models.

$$E_1 = \{e_1, pre_1 = (b_1 \wedge \neg b_2) \vee (\neg b_1 \wedge b_2)\}$$
$$E_2 = \{e_2, pre_2 = (c_1 \wedge \neg b_2) \vee (\neg c_1 \wedge b_2)\}$$

We can map the preconditions to the tree structure in Figure 4.5.

$$\begin{tikzcd}[column sep=small]
& & & pre_1 \arrow[dll, "gr"'] \arrow[drr, "gr"] & & & \\
& \begin{array}{c} b_1 \wedge \neg b_2 \\ pre_2 \end{array} \arrow[dl, "gr"'] \arrow[dr, "gr"] & & & & \begin{array}{c} \neg b_1 \wedge b_2 \\ pre_2 \end{array} \arrow[dl, "gr"'] \arrow[dr, "gr"] & \\
c_1 \wedge \neg b_2 & & \neg c_1 \wedge b_2 & & c_1 \wedge \neg b_2 & & \neg c_1 \wedge b_2
\end{tikzcd}$$

Figure 4.5: A tree for DEL preconditions

Limitation of the tableaux model is that we cannot get the valuation directly from this tree. However, if we apply the initial epistemic model to this tree structure, we can indeed get the correct answer directly.

The initial epistemic world for this game item is $\mathbf{S}_0 = \{S_0, ||\cdot||, s^*\}, |S_0| = 3^2 = 9$. There are 9 possible worlds representing 9 possible flower configurations in the model.

$$\begin{array}{lll} s_1 \models a_1 \wedge a_2, & s_2 \models a_1 \wedge b_2, & s_3 \models a_1 \wedge c_2, \\ s_4 \models b_1 \wedge a_2, & s_5 \models b_1 \wedge b_2, & s_6 \models b_1 \wedge c_2, \\ s_7 \models c_1 \wedge a_2, & s_8 \models c_1 \wedge b_2, & s_9 \models c_1 \wedge c_2. \end{array}$$

Figure 4.6 shows the mapping of possible words as valuations in the tableaux tree.

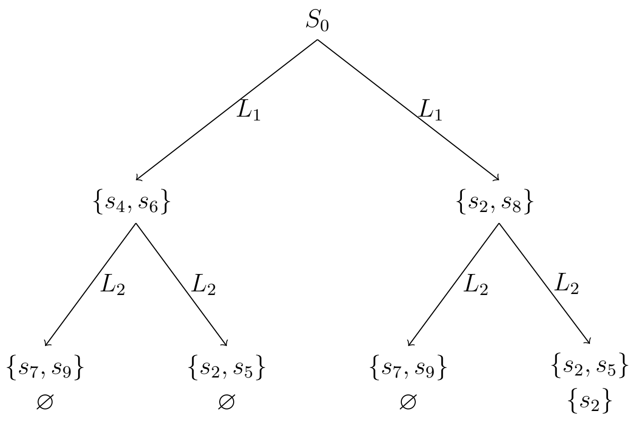
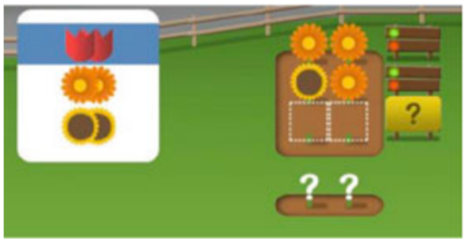


Figure 4.6: A tree for possible worlds

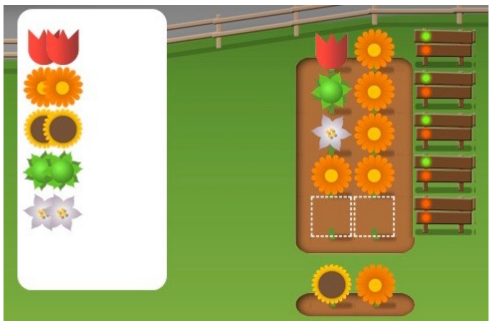
At the end nodes, a possible world is kept if it is included by all the nodes along the branch. In Figure 4.6, only s_2 at the rightmost branch is kept, corresponding to the correct answer of the game. Hence, by translating Boolean formulae in the tableaux model to DEL preconditions and applying them on the DEL initial epistemic model, we can have a more direct way of finding the *goal* valuation in the tableaux decision tree.

4.7 Logical Shortcuts

The DEL model can also represent cross clue reasoning patterns, such as the all *gr* feedback pattern. Recall the game item in Figure 4.7b (same as Figure 3.4) has four clues, each with a *gr* feedback and the same kind of flower appears in the same position in each of them. Some children recognize this pattern and point out that the same flower should appear at the same position in the answer as well. As mentioned earlier the tableaux model cannot represent this kind of cross clue pattern because a decision tree always branches clue by clue. However, the DEL model provides a nice account for such cross clue patterns. Since this cross clue pattern makes a complicate game item easier, we call it a "logical shortcut", or simply a "shortcut", throughout this section.



(a) Game item with two *gr* feedbacks



(b) Game item with four *gr* feedbacks

Figure 4.7: Screenshots of two game items with *gr*-only feedbacks

In the DEL formalization, the *gr* shortcut discussed above is a result of some propositional logic reasoning. Consider the game item in Figure 4.7a (same as Figure 4.2). Let $G = (Fl_G, \mathcal{L}_G, goal), Fl_G = \{a, b, c\}, \mathcal{L}_G = \{L_1, L_2\}, L_1 = C_1 \wedge \sigma_1 = b_1 \wedge b_2 \wedge gr$, and $L_2 = C_2 \wedge \sigma_2 = c_1 \wedge b_2 \wedge gr$.

Let $\mathbf{S}_0$ be the initial epistemic model, $\mathbf{E}_1 = \{e_1, pre\}$, where e_1 is the action of observing clue L_1, and $\mathbf{E}_2 = \{e_2, pre\}$, where $e_2 \in E_2$ is the action of observing clue L_2.

By the product update, $\mathbf{S}_0 \otimes \mathbf{E}_1 \otimes \mathbf{E}_2 = \mathbf{S}_2$, such that for all $s \in S_2, s \models pre_{e_{L_1}} \wedge pre_{e_{L_2}}$. By the definition of preconditions, for all $s \in S_2$ we have:

$$s \models [(b_1 \wedge \neg b_2) \vee (b_2 \wedge \neg b_1)] \wedge [(c_1 \wedge \neg b_2) \vee (b_2 \wedge \neg c_1)]$$

By propositional logic we have:

$$s \models (b_1 \wedge \neg b_2 \wedge c_1) \vee (b_2 \wedge \neg b_1 \wedge \neg c_1)$$

By the definition of flower configurations in Section 4.5, $b_1 \wedge \neg b_2 \wedge c_1$ cannot be a flower configuration, because both b_1 and c_1 are indexed with $\cdot_1$.

Hence, $s \models b_2 \wedge \neg b_1 \wedge \neg c_1$. Therefore, $s \models b_2 \wedge a_1$ and by Theorem 4.1, $goal = b_2 \wedge a_1$.

We can generalize the *gr* shortcut in Theorem 4.3, and prove it based on our DEL formalization and propositional logic rules.

Theorem 4.3 (*gr*-shortcut)**.** In a 2-pin game item where $G = (Fl_G, \mathcal{L}_G, goal)$, and $\mathcal{L}_G = \{L_1, \ldots, L_n\}$, for any $L, L' \in \mathcal{L}_G$, if $L = p_1 \wedge q_2 \wedge gr$ and $L' \in \mathcal{L}_G = r_1 \wedge t_2 \wedge gr$, then

$$s* \models \begin{cases} p_1 \wedge \neg q_2 \wedge \neg t_2, & \text{if } p = r \\ q_2 \wedge \neg p_1 \wedge \neg r_1, & \text{if } q = t \end{cases}$$

Proof. Assume that $p = r, q \neq t$, then $L = p_1 \wedge q_2 \wedge gr$, and $L' = p_1 \wedge t_2 \wedge gr$. Accordingly, we can build two event models $\mathbf{E} = \{e, pre\}$, where $e \in E$ is observing clue L and $\mathbf{E}' = \{e', pre\}$, where $e' \in E'$ is observing clue L'. Let $\mathbf{S}$ be the epistemic state model after updating with both $\mathbf{E}$ and $\mathbf{E}'$, by the definition of product update, for all $s \in S, S \in \mathbf{S}, s \models pre_e \wedge pre_{e'}$. By definition of preconditions, for all $s \in S, S \in \mathbf{S}$ we have:

$$s \models [(p_1 \wedge \neg q_2) \vee (q_2 \wedge \neg p_1)] \wedge [(p_1 \wedge \neg t_2) \vee (t_2 \wedge \neg p_1)]$$

By propositional reasoning we have:

$$s \models (p_1 \wedge \neg q_2 \wedge \neg t_2) \vee (q_2 \wedge \neg p_1 \wedge t_2)$$

By the definition of flower configurations we have:

$$s \models p_1 \wedge \neg q_2 \wedge \neg t_2$$

Note that $q \neq t$ because $p = r, \sigma = \sigma'$ and $L \neq L'$.
By theorem 4.1, $s^* \in S$. Hence, $s^* \models p_1 \wedge \neg q_2 \wedge \neg t_2$ if $p = r$.
Similarly, $s^* \models q_2 \wedge \neg p_1 \wedge \neg r_1$ if $q = t$. □

As discussed above, the *gr* shortcut can be generalized to more than two clues. Let $\mathcal{L}_G$ be a set of clues, and each $L_k \in \mathcal{L}_G$ be a sentence of the form $p_1 \wedge x_2 \wedge gr$, where p is the same flower for all $L_k \in \mathcal{L}_G$ but x is different for all $L_k \in \mathcal{L}_G$. In this game, $s^* \models p_1 \wedge \bigwedge \neg x_2$, x appears in L_k for all $L_k \in \mathcal{L}_G$. A similar rule holds for clues of the form $x_1 \wedge p_2 \wedge gr$ where p_2 stays the same across clues and x changes flowers among clues. We call this the generalization of the *gr* clue.

For example, look at the game item in Figure 3.4. Let the $Fl_G = \{a, b, c, d, e\}$ from top to bottom, the four clues in the game are

$$\begin{aligned} L_1 &= a_1 \wedge b_2 \wedge gr, \\ L_2 &= d_1 \wedge b_2 \wedge gr, \\ L_3 &= e_1 \wedge b_2 \wedge gr, \\ L_4 &= b_1 \wedge b_2 \wedge gr. \end{aligned}$$

By the generalization of *gr* rule,

$$s^* \models b_2 \wedge \neg(a_1 \vee d_1 \vee e_1 \vee b_1)$$

Therefore,

$$s^* \models b_2 \wedge c_1$$

Hence, $goal = b_1 \wedge c_2$.

The ability to represent logical shortcuts in the DEL model also increase its prediction power on item ratings. We can select a group of game items for which shortcuts are applicable, and generate a simpler update sequence for these game items according to the shortcuts. These simplified update sequences will have smaller complexity measurement values when compared to the original (unordered) update sequence, which corresponds to lower ratings of these game items which better match the ratings observed in the empirical dataset. Therefore, incorporating logical shortcuts in the model results in a finer categorization of game items, and is expected to improve the prediction performance of our formalization.

Chapter 5

Empirical Evaluation of Models

In the previous chapter we presented a DEL formalization of DMM game and in this chapter we evaluate this DEL model with regard to the empirical Elo rating data of DMM game items. We test each DEL measurement defined in the previous chapter against the empirical dataset, and compare the performances of the DEL and tableaux models with respect to this same dataset. We show that DEL measurements that take feedback types into consideration perform as well as the tableaux measurements. Presumably the DEL and the tableaux measurements capture the same aspect of the cognitive difficulty of DMM game items because of their similar prediction power, together with the high correlation between the predictions of both formalizations.

5.1 DEL Measurements

In this section, we test the DEL complexity measurements on the prepared dataset. See Table 4.1 for the list of measurements we test. We compute each measurement for the 355 2-pin DMM game items and use the multiple linear regression approach to check how well DEL measurements correlate with the empirical Elo rating data. See appendix for a description of the treatment of the dataset.

The multiple linear regression approach analyzes how well various independent variables explain a dependent variable. The relationship between the dependent variable and a set of independent variables is modeled via a linear predictor function. Each independent variable is associated with a model parameter, and the linear predictor function estimates unknown model parameters from the data (Freedman, 2009). The model parameters indicate how an independent variable effect the dependent variable, and the linear predictor function generates a line that tries to fit the data as much as possible. After the function is computed, a total least square (TLS) method is applied to measure how much the set of independent variables deviates from the linear predictor function, and this deviation is taken as a measurement of how well the set of independent variables explain the dependent variable. In our case, the Elo ratings for game items is the dependent variable that needs to be explained, and we check various features of the game as well as different complexity measurements to see how well they explain the empirical Elo ratings.

5.1.1 Sequential Update

Table 5.1 lists the regression results for the four DEL measurements SUM_0, SUM_1, SV and CR on the update sequences generated by top-to-bottom order. To test these four measurements, we consider five linear models.

Model 0 considers basic features of a 2-pin DMM game item, which include the following: `colors` refers to the number of flower types available for the item, `rows` refers to the number of rows of clues in the game, and `allcolin` tells whether all the types of flowers are used in the clues (`allcolin` is a Boolean value). Model 0 provides a baseline for comparing measurements that use different aspects of solving a game item. According to Tabel 5.1, `rows` and `allcolin` contribute significantly to the prediction, and Model 0 explains 26.79% of the variance.

Table 5.1: Regression results for four DEL measurements

	Model 0	Model 1	Model 2	Model 3	Model 4[9]
Definition		$\sum_{i=0}^{n} S_i$	$\sum_{i=1}^{n} S_i$	$(\sum_{i=0}^{n} S_i)/n$	$(\sum_{i=1}^{n} \frac{S_{i-1}}{S_i})/n$
(Intercept)	−17.8894***	−37.2518***	−29.6852***	−27.7091***	−42.2720***
colors	−0.3354	8.6411***	4.7626***	20.7143***	−5.7883***
rows	4.5300***	6.3047***	6.0788***	−4.5951***	58.0237***
allcolin	−8.8332***	−3.7360***	−3.7666***	−6.3370***	−6.4818***
SUM_0		−0.4843**			
SUM_1			−0.4237*		
SV				−3.3568***	
CR					−75.8919***
R^2	0.2679	0.179	0.171	0.3581	0.425
Adj. R^2	0.2616	0.1696	0.1616	0.3508	0.4184

The four DEL measurements listed in Table 5.1 are Model 1 to Model 4. Model 1 extends Model 0 with the SUM_0 measurement that sums the number of possibilities for solving a game item; Model 2 extends Model 0 with the SUM_1 measurement which leaves out the size of the initial epistemic model from SUM_0; Model 3 extends Model 0 with the SV measurement which takes the average of the SUM_0 measurement for each game item; and Model 4 extends Model 0 with the CR measurement that considers the convergence

[9]Another measurement where each S_i comes from $S_0 \otimes E_{L_i}$ gives similar results.

rate of solving a game item. Each measurement is computed according to its definition for each game item.

The regression analysis shows that each of the four DEL measurements contribute significantly. The basic features, i.e., `colors`, `rows`, `allcolin`, in Model 1 to Model 4 contribute significantly, with the feature `colors` changing from not significant in Model 0 to significant in each of the other four models. This indicates that the number of flowers types used in a game model plays an important role when considering the logic structure for solving a game item.

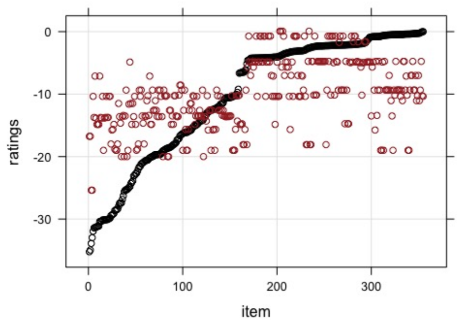
The four DEL measurements have different power in predicting item ratings. Model 1 and Model 2 performs relatively bad, while Model 4 performs the best among the four DEL measurements. Specifically, Model 1 has R^2 of 17.9% and Model 2 has R^2 of 17,1%. R^2 demonstrates how much the given set of independent variables explain the dependent variable, the higher R^2, the better predictions that model makes. It is also known as the variance explained by the linear predictor function. Hence, Model 1 and Model 2 give poor predictions according to our statistics. Model 3 that averages the SUM_0 measurement with number of clues performs slightly better than Model 1 and Model 2, with 35.81% variance explained. Model 4 can explain up to 42.5% of the variance, demonstrating better prediction power of the convergence rate measurement.

Figure 5.1 shows the plots of observed rating data (black points in the plots) and predicted ratings based on the four DEL measurements (red points in the plot). It shows that predictions computed based on the four DEL measurements tested so far still deviate quite a lot from the real situations, as the red predicted dots scatter loosely around the black observation dots.

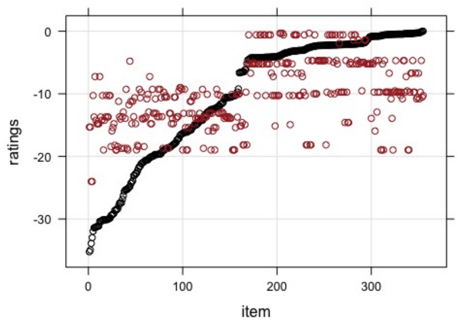
Analysis of update sequence

Table 5.1 shows that the DEL measurements tested so far cannot predict the ratings of items very well. One reason for the poor predictions is that none of the four DEL measurements can categorize game items that have similar Elo ratings under the same complexity measure. Ideally, we want a positive correlation between Elo ratings of a game item and its corresponding DEL complexity measurement. If a game item has a more complex DEL structure, then the game item is expected to be more difficult in practice, resulting in a high Elo rating, and vice versa. However, as we are going to show in the following analysis, these four DEL measurements cannot categorize empirically easy or hard game items well enough.

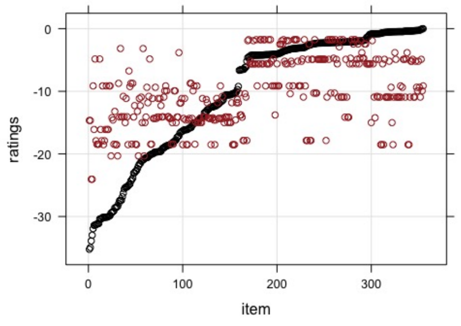
We analyzed the SUM_0 measurement, introduced a method to evaluate how well a DEL measurement categorizes Elo ratings, then we apply this method to all of the four DEL measurements, and discovered that these measurements all fail to categorize Elo ratings of game items well enough.



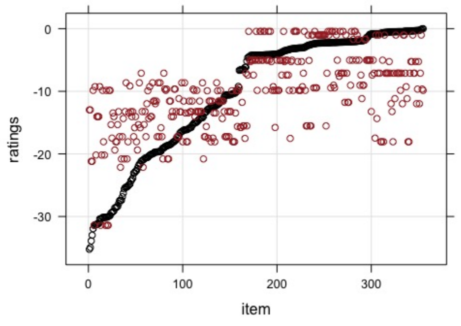
(a) SUM_0



(b) SUM_1



(c) SV



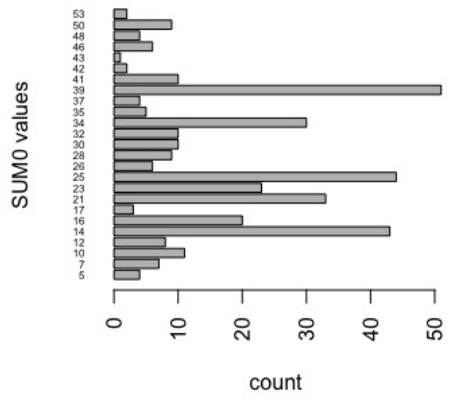
(d) CR

Figure 5.1: Plots for predictions and observed ratings of 2-pin game items

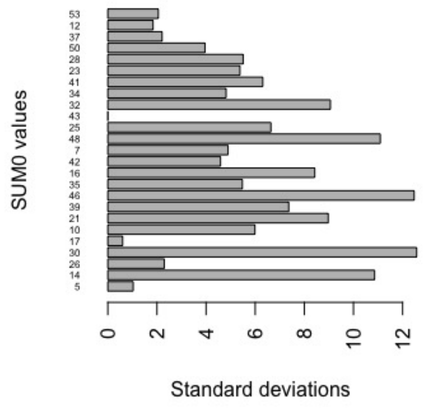
To understand how well the SUM_0 measurement categorizes Elo ratings of the 355 game items, let us first list all the distinct values in the SUM_0 measurement. These distinct values are referred to as SUM_0 values throughout this chapter. There are 25 SUM_0 values out of the 355 game items, and Figure 5.2a shows the count for each SUM_0 values. The distribution for count of SUM_0 values differs quite significantly with the distribution of Elo ratings of 2-pin DMM game items in Figure 2.3, showing a potential mismatch between them.

Let's take the SUM_0 value 14 for a further look. SUM_0 value 14 covers game items with Elo ratings from -31.9436 to -0.0013, with a mean value of -11.69646 and the standard-deviation of 10.85946. Within the scale from -31.9436 to -0.0013, a standard deviation of 10.85946 shows that item ratings spread quite widely. Ideally, one DEL measurement value should capture a class of game items that share similar Elo ratings, so the standard-deviation should be as small as possible. The large standard deviation values shows that the SUM_0 value 14 fails to categorize a class of empirically easy game items.

Calculating standard deviations for all the unique values in the SUM_0 measurement, we get the list in Figure 5.2b. Figure 5.2b shows that though some values in the SUM_0



(a) Count of SUM_0 values



(b) σ for SUM_0 values

measurement can indeed categorize a class of game items that are empirically of similar difficulty level, most of the SUM_0 values fail to do so, with standard deviation $\sigma > 2$. The average standard deviation $\sigma_{mean} = 5.775954$ further confirms that in general SUM_0 values cannot categorize empirical ratings in a neat way.

Table 5.2 lists the standard deviations for all of the four DEL measurements. We can see that none of the four DEL measurements can actually categorize the empirical ratings for DMM game items. This explains why these four DEL measurements do not perform well in the regression model, because each of their values fails to categorize game items with similar ratings.

Table 5.2: Summary for standard deviation σ of distinct values in each DEL measurement

	SUM_0	SUM_1	RV	CR
count	25	15	30	27
min.	0.6031	2.055	0.5828	0.5828
max.	12.5700	12.470	12.5700	14.3100
mean	5.775954	7.197	6.1010	6.0590

5.1.2 Intersecting Update

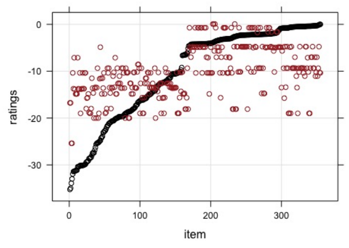
We apply the above mentioned four DEL complexity measurements on ordered update sequences as well, and the the regression results for the recomputed measurements are listed in Table 5.3. Model 1' recomputed SUM_0 values for the unordered DEL structure, Model 2' recomputed the SUM_1 values and Model 3' recomputed the SV values. The regression result shows that Model 1 behaves quite similar to Model 1, explaining 17.22%

of the variance, Model 2 similar to Model 2', explaining 16.75% of the variance, and Model 3' similar to Model 4 in table 5.1, with 27.05% variance explained.

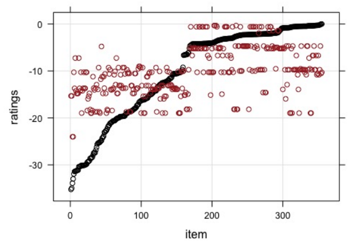
Table 5.3: Regression results for unordered DEL structures

	Model 1'	Model 2'	Model 3'
(Intercept)	−37.7607***	−32.7578***	−37.2804***
colors	7.5445***	5.1031***	18.6707***
rows	7.2826***	7.0439***	0.3659
allcolin	−3.8006***	−3.8159***	−5.4563***
SUM_0	−0.3004**		
SUM_1		−0.2746*	
SUM_0 / rows			−2.5745***
R^2	0.1722	0.1675	0.2705
Adj. R^2	0.1627	0.158	0.2622

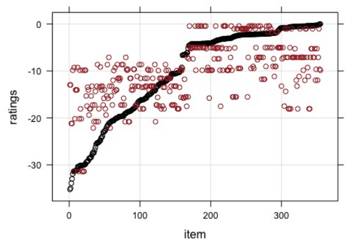
Figure 5.3 compares plots for predicted and observed rating data for each 2-pin game items between normal DEL structures and unordered DEL structures. From the plots, it is clear that neither of the definitions align with observations well enough, and they show very similar distributions. Therefore, either (1) order of processing clues does not really matter for fitting the DEL model with empirical ratings, or (2) there are other factors that influence how logical structures fit with empirical rating data.



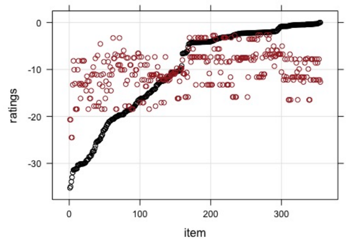
(a) SUM_0



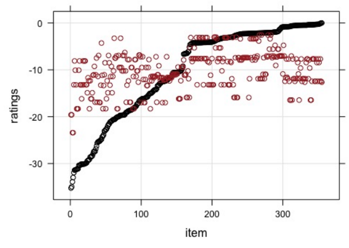
(b) SUM_1



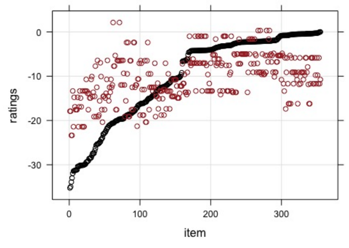
(c) SV



(d) SUM_0'



(e) SUM_1'

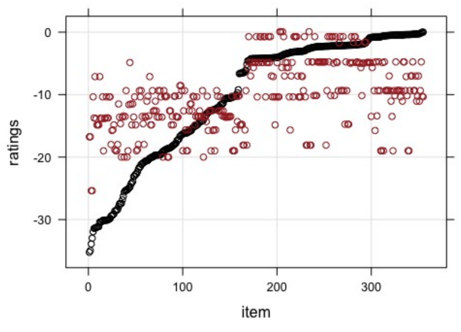


(f) SV'

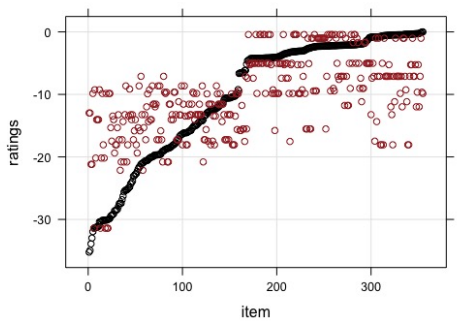
Figure 5.3: Plots for default and unordered predictions

5.1.3 Update per Feedback Type

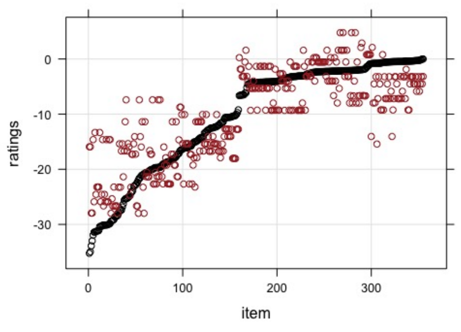
Next, we test the DELs and DELr measurements that take feedback types into consideration. Recall that DELs and DELr measurements compute the average size of updated model, or the average ratio of updated model against initial model, per feedback type. Table 5.6 shows the regression result of DELs and DELr based on Elo ratings of 355 game items. Compared to Table 5.1, both DELs and DELr improve the fitting of DEL measurements with data and can explain up to 63% and 67% of the variance. This shows that considering the type of feedbacks can predict item ratings of 2-pin DMM game items quite well. This result is intuitive because people process feedback to solve the game, and different feedbacks pose different computational problems. For example, the *gr* excludes a flower and confirms another one, and the *or* feedback does not only exclude a flower, but also says that the other flower is of the wrong position. These two feedbacks may influence the time and cognitive resource an agent need to reason about them. Statistical result further confirms that feedback is an important predictor of the cognitive difficulty of DMM game items.



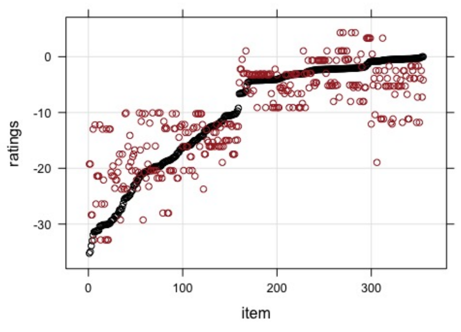
(a) SUM_0



(b) SV



(c) DELs



(d) DELr

Figure 5.4: Plots for DEL measurements that consider feedback types

Figure 5.4 further shows the improvement of prediction. Figure 5.4a and Figure 5.4b are predictions of item rating based on the SUM0 and RS measurements, which as presented above does not align with the observed data. Figure 5.4c and Figure 5.4d are predictions of item rating based on the DELs and DEr measurement. As shown from the plots, the red predicted values center around the black observed rating data much closer than the the two plots above, displaying a clear improvement of prediction accuracy. The DELs and DELr plots shows two clear clusters among easier items and harder items, with the harder items aligning even more closely to the observed rating data. This indicates that feedback types does play an important role in predicting harder game items. Moreover, in Figure 5.4a and Figure 5.4b , predicted values can hardly match with some of the easiest items, but in Figure 5.4c and Figure 5.4d is problem is solved. This indicates that considering feedback types also helps to make better predictions concerning the easiest game items.

5.2 Comparing Tableaux and DEL Measurements

In this section, we compare how well the tableaux and DEL models explain Elo ratings. Firstly, we tested the tableaux measurements on the dataset for May 2017, and discovered some noteworthy variances when compared to Gierasimczuk et al. (2013). Figure 5.5 shows the plot of item rating distribution for the 2013 and 2017 dataset. The figure on the left is replicated from Gierasimczuk et al. (2013), which shows a clear bimodal distribution of 2-pin item rating data. The figure in the middle is the distribution of 2-pin item ratings up to May 2017, and the bimodal distribution is not clear anymore. Item ratings peak around -2.5, and ratings lower than -10 have a more uniform distribution. As mentioned before, in 2013 the number of game items was 100, and in 2017 the number is 355. Therefore, we picked out from the 2017 dataset the game items that corresponded to the 100 items in the 2013 dataset and plotted their item rating distribution, which is the plot on the right in Figure 5.5. It shows a transition from the bimodal distribution to the pattern of peak and uniform. There are many possible reasons for this change. One is that there is a new version of game display between 2013 and 2017, and the designer changed the way how feedback looks like on the screen. In the old version, feedbacks were all presented horizontally, and in the new version they are presented vertically.

We then tested tableaux methods on the entire 2017 dataset and listed the results in Table 5.4. Model 1 and Model 2 are from Gierasimczuk et al. (2013), measuring application steps for each feedback based on the default and least decision tree. On the 2013 dataset, Model 1 explains 75% of the variance, and Model 2 explains 70% of the variance. We computed the same measurement for the 355 game items in the 2017 dataset, and listed them as Model 1' and Model 2' in Table 5.4. Model 1' explains 62.53% of the variance and Model 2' explains 66.14% of the variance. The performance of both of the

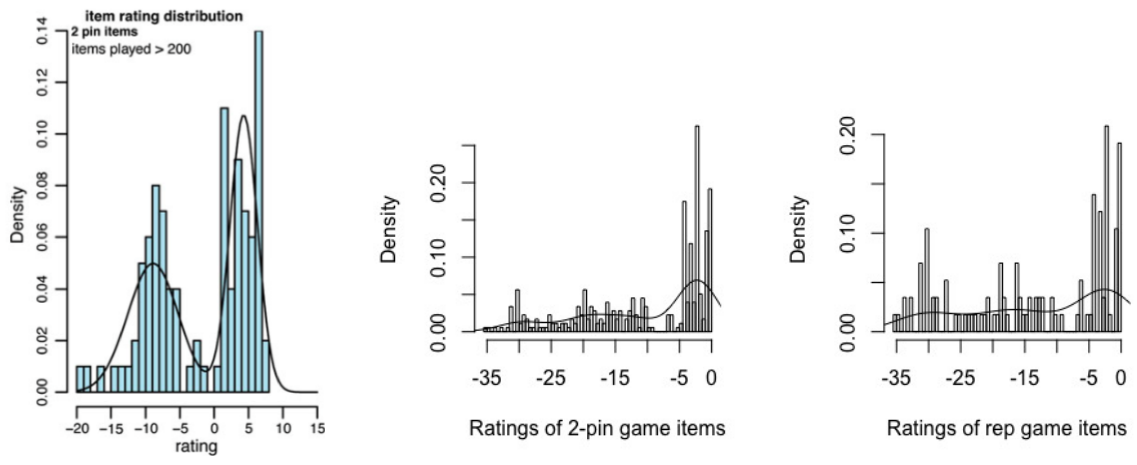


Figure 5.5: Plots for ratings of 2-pin game items

new models are less than the 2013 result, but are still good enough in terms of predicting item ratings.

Table 5.4: Regression results for tableaux models on the old and new datasets

	Model 1	Model 2	Model 1’	Model 2’
(Intercept)	−7.55***	−8.64***	−15.8704***	−14.432994***
colors	0.92	0.57	3.7394***	4.562682***
rows	3.16***	2.47***	−0.9797	−1.016774*
allcolin	−5.83***	−6.00***	−6.4096***	−6.089445***
oo	−2.47*	−0.41	−6.9381***	−11.455507***
rr	−3.56***	−1.94***	−6.9381	−2.983262***
gr	0.12	0.45	0.6605**	0.003135
or	2.23	2.47***	3.0772***	2.518258***
R^2	0.75	0.70	0.6253	0.6614
Adj. R^2	0.73	0.68	0.6178	0.6545
Num. obs.	100	100	355	355

Next we compared the predictions from the tableaux and DEL models based on the 2017 dataset. Table 5.5 lists the regression results for the two DEL measurements with feedback types and the results from the analytical tableaux models. In Table 5.5, `oo, rr, gr, or` variables for model `Tableaux` and `Tableaux’` are steps for the corresponding feedbacks in tableaux decision trees, and `oo, rr, gr, or` variables for model `DELs` and `DELr` are number of possible worlds eliminated per feedback and convergence rate per feedback as defined in Section. Table 5.5 shows that the models based on tableaux decision trees and DEL models that incorporate feedback types have similar statistical performances. All four of the models explain more than 60% of the variance, with Model `DELr` performing the best, explaining 67.2% of the variance, and Model `Tableaux` performing the worst, but still explaining 62.53% of the variance.

Table 5.5: Comparing tableaux model with DEL measurements

	Tableaux	Tableaux'	DELs	DELr
(Intercept)	−15.8704***	−14.432994***	−10.28256***	−26.8712***
rows	3.7394***	4.562682***	2.47430***	34.6568***
colours	−0.9797	−1.016774*	−2.16959***	−7.6840***
allcolin	−6.4096***	−6.089445***	−5.02605***	−6.9753***
oo	−6.9381***	−11.455507***	−8.98402***	−55.2099***
rr	−6.9381	−2.983262***	−0.04604	−46.6109***
gr	0.6605**	0.003135	0.70094***	−41.0523***
or	3.0772***	2.518258***	1.83475***	−21.3221***
R^2	0.6253	0.6614	0.6322	0.672
Adj. R^2	0.6178	0.6545	0.6248	0.6654

Table 5.5 shows that the tableaux and DEL models share similar statistical performances on the same dataset. What if we combined the two methods to create a richer model to predict item difficulty? Usually, incorporating more parameters in a regression model leads to better data fit. For example, if Model A uses 2 parameters to predict weather, and Model B uses 20, then Model B is more likely to give a precise prediction of the weather, because it takes more factors into consideration. Following this reasoning, if we merge the tableaux and DEL models into one we should get better predictions about the difficulty of an item.

However, Table 5.6 shows that merging the two models does not produce better prediction of item difficulty. Table 5.6 lists three models, one is the tableaux model with halting condition on the least tree, which has the best statistical performance among all tableaux models. Another is the DELr model that measures convergence rate with respect to feedback types, which also gives the best prediction for item difficulty among all models that incorporate DEL measurements. A combing model of these two models is denoted as `Tableaux + DEL` in Table 5.6, which contains independent variables of the basic game features, measurements commutated by the tableaux model, and measurements commutated by the DEL model. Since we are only interested in how well each model predicts rating of game items, we omit the estimated parameters for independent variables in the table, and focus on the R^2 value of each model. Surprisingly, even though have more parameter, the combined model performs just slightly better than the individual tableaux or DEL model, with 67.2% variance explained.

A further analysis on the correlation of the tableaux and DEL models reveals that the two share very high correlations with each other. This result is surprising because the

Table 5.6: Regression results for the analytical tableaux model and combined models

	Tableaux (least tree)	DELr	Tableaux + DEL
(Intercept)	-14.432994^{***}	-10.28256^{***}	-10.1257^{***}
rows	4.562682^{***}	2.47430^{***}	-2.3511^{***}
colours	-1.016774^{*}	-2.16959^{***}	1.7511^{*}
allcolin	-6.089445^{***}	-6.089445^{***}	-5.4334^{***}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
R^2	0.6614	0.672	0.6847
Adj. R^2	0.6545	0.6654	0.6746

two models are based on different formalizations of the game, and each measures different aspects of game solutions. The tableaux model focuses on case-switching, whereas the DEL model cares about number of possible options and convergence rate. The tableaux method computes application steps while searching over the decision tree, whereas the DEL model counts the number of possible worlds or the ratio of selected worlds against the initial epistemic world with each feedback type. However, the two models still display similar statistical resemblance and high correlation, which indicates that the two models capture similar aspects of the cognitive difficulty of DMM game items, despite their differences in the modeling details.

A possible explanation for this result is that the influence of feedback types of item difficulty is so strong that it is not effected by what logic one uses to model the game. Recall that in the tableaux model, size of a decision tree is represented by application steps of each feedback types, and in the DELs and DELr model, complexity is measured as average number or ratio of selected possible worlds per feedback type. Similar regression results imply that it is the feedback type that strongly influences the prediction of item ratings, and high correlation among the two logic formalizations also confirms that they are both measuring something similar about game. DEL models that do not break down with respect to feedback types cannot predict item ratings as well as DEL models that

Table 5.7: Correlation analysis

	dOO	dRR	dGR	dOR
tOO	0.9642	-0.0763	-0.2312	-0.2843
tRR	-0.0720	0.7475	-0.0012	-0.2177
tGR	-0.3165	-0.0580	0.6448	-0.1270
tOR	-0.2674	-0.1293	-0.2096	0.7632

consider feedback types, which indicates that ignoring feedback types is a reason for the poor statistical performances for those models.

Putting all this information together, it is reasonable to conclude that the effect of feedback type is a strong predictor of the cognitive difficulty of the DMM game, and it is prominent enough to be detected with different formalizations. Back to the question we asked earlier about whether the predictions the tableaux model gave are artifacts of the formalization itself, now it seems that the answer is no. That is because the DEL model is a different formalization of the game, and the DEL model has similar statistical results and high correlation compared with the tableaux model.

Chapter 6

Conclusion

In this thesis we studied the cognitive difficulty of the Deductive Mastermind (DMM) game by measuring the complexity of the game in two different logic formalizations. We looked at an existing formalization of DMM based on an analytic tableaux, and we developed a formalization based on dynamic epistemic logic (DEL). We found that feedback types play an important role in predicting cognitive difficulty of game items. This result was robust over the two different logic formalizations that we considered.

DMM is a simplified version of the board game Mastermind, and it has been implemented within a popular online educational game system. In this system, each DMM game item is associated with a rating to reflect its cognitive difficulty based on the accumulated data of how quickly and accurately children solved it. We investigated a DEL formalization of DMM, in which we used epistemic models to represent possible answers, and event models to encode information related to the reasoning about clues. In the game setting, a DEL solution to a game item will always lead to a final epistemic model where only one world is left, namely, the actual world which represents the correct answer of that game. The DEL model of DMM can also account for cross clue reasoning patterns, such as the all *gr* feedback shortcut that children self-reported.

The DEL solution to a 2-pin DMM game item results in a series of epistemic models that can be either order-depend or order-independent. We call such a series of epistemic models an (unordered) update sequence and measure the logical complexity of each update sequence by the sum of the sizes of all epistemic models, the converging rate of an update sequence, or similar measurements that take feedback types into account as well. Testing those complexity measurements with the empirical rating data shows that DEL measurements that do no take feedback types into consideration cannot explain the empirical dataset well, and DEL measurements that do consider such feedback types perform better than the tableaux model. Combining the DEL and tableaux model leads to similar predictions of item ratings, and together with the fact that there is a high correlation between the tableaux and the DEL models, this shows that feedback type is a strong indicator of the cognitive difficulty of a DMM game item. This result also indicates that item ratings are not influenced by the choice of the logic used in developing a formal model of the game.

There is great potential for future work continuing this line of research. I outline three possible directions in the following paragraphs.

1. Error patterns

   One possibility is to extend the DEL formalization to account for the error patterns observed in the empirical dataset. We can try to add and test several hypotheses, such as that agents may arrive at a wrong epistemic model with a certain probability or that they can accidentally exclude or include possible worlds. By playing with these hypotheses, we can test how well the DEL formalization explains errors in playing DMM game items.

2. Atomic operation

   As this thesis reveals, feedback types play a decisive role in predicting the cognitive difficulty of a game item. Another possible direction for future research is to generalize this result to n-pin game items. One way to study such generalizations efficiently may be to consider the role of atomic operations in feedbacks. A green feedback refers to a confirmation of flowers, a red feedback refers to an elimination of flowers, and an orange feedback refers to possibilities of case-analysis. Within the DEL formalization, we can use these three atomic operations to partition the set of possible flower configurations, and study their influence on the logical complexity of more generalized cases.

3. Cognitive model

   The DEL formalization explores different possible ways of solving DMM, either clue by clue or via cross-clue reasoning patterns. These investigations can inspire models that aim to capture the cognitive process of children for solving DMM. For example, we can assume that some children make use of the cross clue reasoning pattern as described by the DEL formalization, and turn it into a process model that assumes the procedure for solving a game item is conducted in certain number of steps and each step takes certain amount of time or cognitive resources. We may also post a threshold for the amount of cognitive resources a play requires in solving the game and try to predict the speed and accuracy performance based on these assumptions.

This thesis is a case study of using the combination of logic and computational analysis to capture cognitive difficulty of solving DMM game items. It analyzes formalizations of the DMM game in different logics, and investigates how different complexity measurements of those logic formalizations predict cognitive difficulty of the game differently.

From a formal perspective, this thesis presents a DEL formalization of the DMM game that makes fewer assumptions than an existing tableaux model about an agent's cognitive process. The DEL formalization is also richer than the tableaux model because it can account for some complicated reasoning patterns observed in children's self-reports of the game. From an empirical perspective, this thesis demonstrates that feedback types for the game is a strong predictor of the cognitive difficulty of the game, and this feature is captured by both logic formalizations tested in this thesis. By comparing two different logic formalizations, this thesis confirms that the intrinsic feature that strongly influences the cognitive difficulty of a task can be prominent enough to be detected with different formalizations. In all, this thesis combines logic and cognitive science, bridges formal and statistical approaches to human reasoning, and successfully applies logic and computational analyses in studying the cognitive difficulty of the DMM game.

Bibliography

A. Baltag. Lecture notes in dynamic epistemic logic, February 2016.

A. Baltag, L. Moss, and S. Solecki. The logic of public announcements and common knowledge and private suspicions. In *Proceedings of the 7th Conference on Theoretical Aspects of Rationality*, pages 43–56, 1998.

E. Barton, R. Berwick, and E. Ristad. *Computational Complexity and Natural Language.* Cambridge, MA: The MIT Press, 1987.

J. van Benthem. Semantic tableaus. *Nieuw Archief voor Wiskunde*, 22:44–59, 1974.

J. van Benthem, J. van Eijk, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.

R. Berwick and A. Weinberg. *The Grammatical Basis of Linguistic Performance: Language Use and Language Acquisition.* Cambridge, MA: MIT Press, 1984.

E.W. Beth. Semantic entailment and formal derivability. *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen Afdeling Letterkunde*, 18(13):309–342, 1955.

S. Brown. Late to the game: Mastermind @WIRED, 2012. URL `https://www.wired.com/2012/06/late-tothe-game-mastermind/`.

C. Cherniak. *Minimal Rationality.* Cambridge, MA: The MIT Press, 1986.

H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337. Springer Science & Business Media, 2007.

A.E. Elo. *The Rating of Chessplayers, Past and Present.* Arco Pub., 1978.

Z. Fangzhou, J. Szymanik, and I.A. Titov. Toward probabilistic natural logic for syllogistic reasoning. In *Proceedings of the 20th Amsterdam Colloquium*, pages 468–477, 2015.

J. Feldman. Minimization of boolean complexity in human concept learning. *Nature*, 407 (6804):630–633, 2000.

D. A. Freedman. *Statistical models: theory and practice*. Cambridge University Press, 2009.

N. Gierasimczuk and J. Szymanik. Branching quantification v. two-way quantification. *Journal of Semantics*, 26(4):367–392, 2009.

N. Gierasimczuk, H.L.J. van der Maas, and M.E.J. Raijmakers. An analytic tableaux model for deductive mastermind empirically tested with a massively used online learning system. *Journal of Logic, Language and Information*, 22(3):297–314, 2013.

A.M.C. Isaac, J. Szymanik, and R. Verbrugge. Logic and complexity in cognitive science. In *Johan van Benthem on Logic and Information Dynamics*, pages 787–824. Springer, 2014.

S. Klinkenberg, M. Straatemeier, and H.L.J. van der Maas. Computer adaptive practice of maths ability using a new item response model for on the fly ability and difficulty estimation. *Computers & Education*, 57(2):1813–1824, 2011.

H.L.J. van der Maas. Observed cross clue patterns. Thesis meeting, 2017.

H.L.J. van der Maas, S. Klinkenberg, M. Straatemeier, et al. Rekentuin. nl: Combinatie van oefenen en toetsen. *Examens*, (4):10–14, 2010.

G. Maris and H.L.J. van der Maas. Speed-accuracy response models: Scoring rules based on response time and accuracy. *Psychometrika*, 77(4):615–633, 2012.

Mastermind (board game). — Wikipedia, the free encyclopedia, 2017. URL `https://en.wikipedia.org/wiki/Mastermind_(board_game)`. [Online; accessed 29-June-2017].

S.T. Piantadosi, J.B. Tenenbaum, and N.D. Goodman. The logical primitives of thought: Empirical foundations for compositional cognitive models. *Psychological Review*, 123 (4):392, 2016.

E. Ristad. *The Language Complexity Game*. Cambridge, MA: The MIT Press, 1993.

I. van Rooij, J. Kwisthout, M. Blokpoel, J. Szymanik, T. Wareham, and I. Toni. Intentional communication: Computationally easy or difficult? *Frontiers in Human Neuroscience*, 5:1–18, 2011.

C.E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

R. Smullyan. *First-Order Logic*. Berlin: Springer, 1968.

J. Stuckman and G. Zhang. Mastermind is NP-complete. *arXiv:cs/0512049v1 [cs.CC]*, 2005.

J. Szymanik. *Quantifiers and Cognition: Logical and Computational Perspectives*, volume 96 of *Studies in Linguistics and Philosophy*. Springer, 2016.

J. Szymanik and M. Zajenkowski. Comprehension of simple quantifiers: Empirical evaluation of a computational model. *Cognitive Science*, 34(3):521–532, 2010.

G. Truescu. Logical reasoning in a deductive version of the Mastermind game. Master's thesis, Leiden University, 2016.

M. Zajenkowski, R. Styła, and J. Szymanik. A computational approach to quantifiers as an explanation for some language impairments in schizophrenia. *Journal of communication disorders*, 44(6):595–600, 2011.

Appendices

Processing of the Dataset

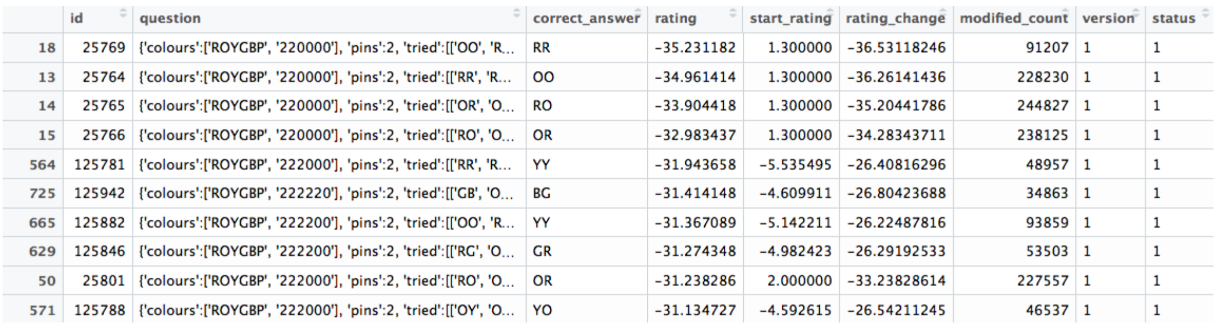
Our dataset is the list of Elo ratings for 2-pin DMM game items. In the 2013 dataset there are 100 game items, by 2017, there are 355 game items in the dataset. The original dataset of 2-pin game item ratings is shown as in Figure 1. As one can see from the figure, the description of a game item is listed as a string that contains all relevant information. For example, game item with id 25769, which is the first line in Figure 1, has a description of the game item (the `question` column) as follows:

`{'colours':['ROYGBP', '220000'],`

`'pins':2, 'tried':[['OO', 'RR']], 'triesLeft':1}`

In every description, `'colours'` shows the available types of flowers for the corresponding game item. In this case, `['ROYGBP', '220000']` means that there there are 2 of the `R` flower and 2 of the `O` flower, and none of the `Y, G, B` or `P` flower. The `'pins':2` string simply says that this is a 2-pin game item. The `'tried'` string lists the clues of the corresponding game item. The first string in a pair of `[...]` strings is the flower configuration, and the second string in the pair is the feedback sequence. For example, in this game item, there is only one clue, with a `OO` flower configuration, and a red-red feedback. The `'triesLeft':1` string marks the end of the question description.

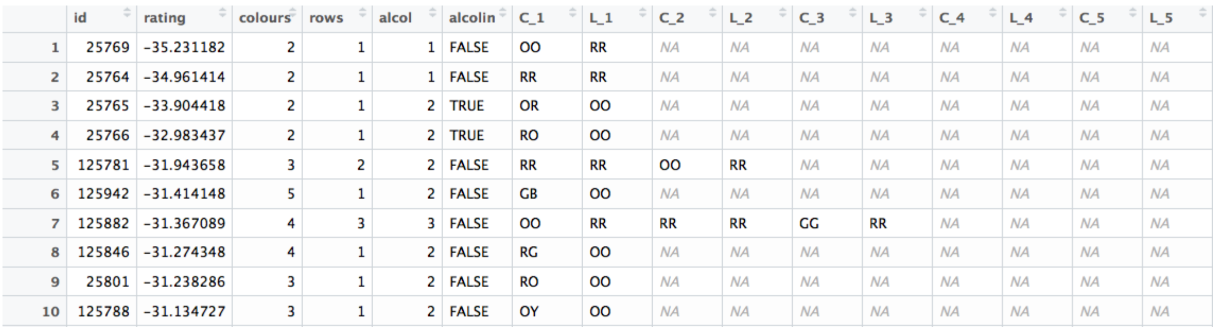
To calculate the DEL measurements of a game item, we need to first structure the dataset in such a way that each flower configuration and feedback is specified and annotated properly. Moreover, we want to compute the basic features of a game item, such as the number of available flower types, number of clues, whether all flowers are used in the



	id	question	correct_answer	rating	start_rating	rating_change	modified_count	version	status
18	25769	{'colours':['ROYGBP', '220000'], 'pins':2, 'tried':[['OO', 'R...	RR	-35.231182	1.300000	-36.53118246	91207	1	1
13	25764	{'colours':['ROYGBP', '220000'], 'pins':2, 'tried':[['RR', 'R...	OO	-34.961414	1.300000	-36.26141436	228230	1	1
14	25765	{'colours':['ROYGBP', '220000'], 'pins':2, 'tried':[['OR', 'O...	RO	-33.904418	1.300000	-35.20441786	244827	1	1
15	25766	{'colours':['ROYGBP', '220000'], 'pins':2, 'tried':[['RO', 'O...	OR	-32.983437	1.300000	-34.28343711	238125	1	1
564	125781	{'colours':['ROYGBP', '222000'], 'pins':2, 'tried':[['RR', 'R...	YY	-31.943658	-5.535495	-26.40816296	48957	1	1
725	125942	{'colours':['ROYGBP', '222220'], 'pins':2, 'tried':[['GB', 'O...	BG	-31.414148	-4.609911	-26.80423688	34863	1	1
665	125882	{'colours':['ROYGBP', '222200'], 'pins':2, 'tried':[['OO', 'R...	YY	-31.367089	-5.142211	-26.22487816	93859	1	1
629	125846	{'colours':['ROYGBP', '222200'], 'pins':2, 'tried':[['RG', 'O...	GR	-31.274348	-4.982423	-26.29192533	53503	1	1
50	25801	{'colours':['ROYGBP', '222000'], 'pins':2, 'tried':[['RO', 'O...	OR	-31.238286	2.000000	-33.23828614	227557	1	1
571	125788	{'colours':['ROYGBP', '222000'], 'pins':2, 'tried':[['OY', 'O...	YO	-31.134727	-4.592615	-26.54211245	46537	1	1

Figure 1: The first 10 items from the raw dataset of 2-pin DMM game items

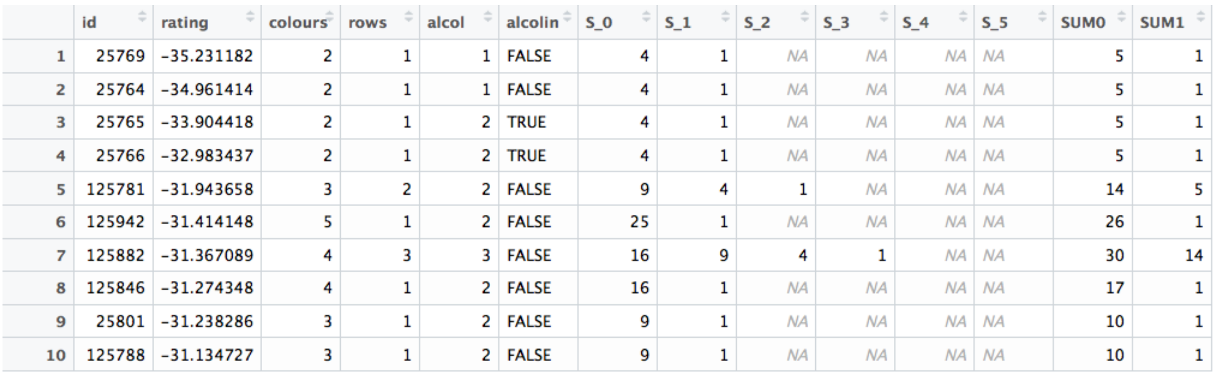
clues, etc. Figure 2 shows the result of such preparation of the dataset. This new dataset keeps only relevant information on testing the DEL measurements, i.e., the rating data, number of available flower types (`colours`), number of clues (`rows`), number of flowers used in the clues (`alcol`), and whether all available types of flowers are used in the clues (`alcolin`). The `question` string is divided into separate columns for flower configuration and feedbacks. In the figure, `C_n` refers to flower configurations, and `L_n` refers to feedbacks.



	id	rating	colours	rows	alcol	alcolin	C_1	L_1	C_2	L_2	C_3	L_3	C_4	L_4	C_5	L_5
1	25769	-35.231182	2	1	1	FALSE	OO	RR	NA	NA	NA	NA	NA	NA	NA	NA
2	25764	-34.961414	2	1	1	FALSE	RR	RR	NA	NA	NA	NA	NA	NA	NA	NA
3	25765	-33.904418	2	1	2	TRUE	OR	OO	NA	NA	NA	NA	NA	NA	NA	NA
4	25766	-32.983437	2	1	2	TRUE	RO	OO	NA	NA	NA	NA	NA	NA	NA	NA
5	125781	-31.943658	3	2	2	FALSE	RR	RR	OO	RR	NA	NA	NA	NA	NA	NA
6	125942	-31.414148	5	1	2	FALSE	GB	OO	NA	NA	NA	NA	NA	NA	NA	NA
7	125882	-31.367089	4	3	3	FALSE	OO	RR	RR	RR	GG	RR	NA	NA	NA	NA
8	125846	-31.274348	4	1	2	FALSE	RG	OO	NA	NA	NA	NA	NA	NA	NA	NA
9	25801	-31.238286	3	1	2	FALSE	RO	OO	NA	NA	NA	NA	NA	NA	NA	NA
10	125788	-31.134727	3	1	2	FALSE	OY	OO	NA	NA	NA	NA	NA	NA	NA	NA

Figure 2: The first 10 items from the structured dataset of 2-pin DMM game items

Based on this structured dataset, we can now construct the DEL models of each game item, and compute the above defined DEL measurements for each item. Figure 3 shows the first 10 items that result from such construction and computation. For each 2-pin game item, based on their update sequence, we compute four DEL complexity measurements SUM_0, SUM_1, RV and CR for them. The DELs and DELr measurements are computed based on unordered update sequence following similar manner. Figure 3 lists only the SUM_0 and SUM_1 measurements, but the others are computed for each game item in a similar manner.



	id	rating	colours	rows	alcol	alcolin	S_0	S_1	S_2	S_3	S_4	S_5	SUM0	SUM1
1	25769	-35.231182	2	1	1	FALSE	4	1	NA	NA	NA	NA	5	1
2	25764	-34.961414	2	1	1	FALSE	4	1	NA	NA	NA	NA	5	1
3	25765	-33.904418	2	1	2	TRUE	4	1	NA	NA	NA	NA	5	1
4	25766	-32.983437	2	1	2	TRUE	4	1	NA	NA	NA	NA	5	1
5	125781	-31.943658	3	2	2	FALSE	9	4	1	NA	NA	NA	14	5
6	125942	-31.414148	5	1	2	FALSE	25	1	NA	NA	NA	NA	26	1
7	125882	-31.367089	4	3	3	FALSE	16	9	4	1	NA	NA	30	14
8	125846	-31.274348	4	1	2	FALSE	16	1	NA	NA	NA	NA	17	1
9	25801	-31.238286	3	1	2	FALSE	9	1	NA	NA	NA	NA	10	1
10	125788	-31.134727	3	1	2	FALSE	9	1	NA	NA	NA	NA	10	1

Figure 3: The first 10 items from the dataset after computing the SUM_0 and SUM_1 measurements