A Unified Framework for Knowledge Assessment and Progression Analysis and Design

Shuhan Wang[1], **Fang He**[2], **and Erik Andersen**[1]

[1]Department of Computer Science
Cornell University
{forsona, eland}@cs.cornell.edu

[2]Department of Computer Science
Peking University
wh27hf@gmail.com

ABSTRACT

Designing engaging learning content is important but difficult, and typically involves a lot of manual specification. We present a unified framework that utilizes automatic problem decomposition and partial ordering graph construction to facilitate multiple workflows: knowledge assessment and progression analysis and design. We present results from a study with 847 participants in an online Japanese-language assessment tool demonstrating that our framework can efficiently measure student ability and predict student performance on specific problems. We also present results from analysis of curricula showing that the progressions of two different textbooks are surprisingly similar, and that our framework can lead to the discovery of general principles of expert progression design. Finally, we demonstrate automatic progression generation with desired sequencing and pacing, allowing for tailoring of progressions and mapping of parameters extracted from one curriculum onto another.

ACM Classification Keywords
H.5.0 Information Interfaces and Presentation: General

Author Keywords
education; automatic problem decomposition; knowledge assessment; progression analysis and design

INTRODUCTION

A key challenge in education is keeping students engaged, and a primary consideration is the optimal sequence of knowledge to present to a particular student. One challenge is ordering: which knowledge should be introduced first? Another is speed: how quickly should the progression get more difficult? Although there are guiding principles from education and psychology [15, 55, 59] that can help design progressions, we lack pragmatic and automatic ways to bridge the gap between educational theory and practice, especially at the scale of an entire curriculum. The difficulty of designing and optimizing progressions also limits the ability of curricula to adapt to the needs of each student.

CHI 2017, May 06 - 11, 2017, Denver, CO, USA

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-4655-9/17/05...$15.00

DOI: http://dx.doi.org/10.1145/3025453.3025841

In HCI, this problem is relevant to the design of educational software [7, 14], video games [11, 22, 33, 37, 45], and software learnability [25, 31]. We currently design progressions mostly by hand, which is expensive and limits our ability of experimenting to find the best progressions. Recently, there has been some work in automatic progression design [6, 11]. Although this work is capable of organizing content automatically, it only addressed procedural topics and video game level design. We still lack a general framework of characterizing educational progressions capable of handling some non-procedural topics like language learning. Furthermore, we need a formal way to assess a student's knowledge using the same framework, so that we can tailor the learning progression and pace for that student. Ideally, we would be able to build adaptive progressions for a wide range of educational domains, with a minimum of effort.

In this paper, we present a framework that allows for automatic problem decomposition and partial ordering graph construction for non-procedural tasks. This framework facilitates multiple workflows. It builds upon item response theory [53], knowledge space theory [28], and knowledge tracing [14] to create multidimensional metrics of student knowledge and problem difficulty, which allows for rapid assessment and prediction of performance. It also defines a parameter space of pacing and other progression parameters, and provides a way to characterize and compare progressions as well as design new progressions that conform to desired parameters.

We present multiple evaluations of our framework. First, we present theoretical contributions for automatic problem decomposition and partial ordering graph construction for grammatical understanding in a foreign language, an important non-procedural topic. We then discuss a user evaluation of a Japanese language assessment tool with 847 participants, demonstrating that our automatic assessment algorithms can lead to accurate measurement of ability and predictions of performance according to item response theory. We also present results for automatic analysis of language learning progressions, showing that two textbooks for learning Japanese have surprisingly similar pacing and other progression characteristics, and that two online language learning tools have very different progression design strategies. Finally, we demonstrate that our framework can automatically generate progressions with desired characteristics, such as the learning pace and the proportion of learning and review. This allows for insights gained from progression analysis to be directly applied to progression design and individual progression tailoring.

RELATED WORK

Intelligent Tutoring Systems

There are successful adaptive learning systems such as Cognitive Tutors [7]. Some of this work has focused on language learning specifically [63, 65]. Cognitive Tutors utilize knowledge tracing [14] to track knowledge acquisition and provide tailored instruction, by tracking performance on individual production rules in a cognitive model [14, 39]. This model has been extended in several ways, including estimating of the initial probability that the student knows a skill [49], estimating of the impact of help features on probability of acquisition [10], and integrating with models of item difficulty [50]. Another method is logistic regression, which is particularly efficient for tasks involving multiple skills [64]. However, these approaches typically do not consider pacing. Instead, they continually give problems that exercise a specific production rule until a Hidden Markov Model has achieved 95% confidence that the student has learned that rule, and then move on to the next concept. Furthermore, Cognitive Tutors are difficult to construct. It has been estimated that as much as 200-300 hours of expert design effort are required to design a single hour of content [3], although newer design techniques have reduced this to 50-100 hours [4]. Our work seeks to develop automatic ways of analyzing and optimizing educational progressions.

Education and Psychology

Many theories support the idea that *moderate difficulty* is important for engagement [1, 8, 16, 20, 34, 47]. Chess players are happiest when playing players ranked slightly above them [1]. Analyses of progressions in non-competitive games also support the use of moderate difficulty [44]. Csikszentmihalyi's theory of flow stipulates that people are most engaged when the challenge is neither too high or too low [16]. Vygotsky's zone of proximal development [59] stipulates that there is a set of concepts that a learner can acquire next with some guidance, and educational content should continually target this set. Reigeluth and Stein's Elaboration Theory [55] argues that the simplest version of a task should be taught first, followed by progressively more complex tasks that elaborate on the original task. Li et al. studied problem orderings by examining them with a machine learning agent, finding that interleaved problem orderings led to faster learning than blocked orderings [43]. We currently lack pragmatic and automatic ways to quantitatively describe these principles and apply them to real progressions, and our work aims to do this.

Language education

Krashen [42] proposed that concepts are learned in a predictable order (the *natural order hypothesis*), and that language acquisition occurs when a learner is exposed to language that is just beyond the edge of what he or she can understand (the *input hypothesis*). There is widespread evidence in the language learning literature that learners progress through distinct stages in their use of grammatical morphemes [30], verb negation [13, 62], questions [52], possessive determiners [61, 66], relative clauses [29, 38], and temporal reference [48]. Our work seeks to create a platform that can evaluate these theories and determine if they can motivate grounded learning progressions that lead to increased learning and engagement.

Trace-based partial orderings

Andersen et al. proposed a technique for automatically exploring the space of task progressions through static analysis of the procedure to be learned [6]. This technique characterizes tasks by analyzing the *execution trace* obtained by running the procedure on that task. By characterizing each task as a sequence of basic operations, one can specify a *partial ordering* that ranks the difficulty of problems. This partial ordering has been experimentally confirmed to match well with users' perceptions of difficulty in an educational algebra game [6]. The trace-based framework has been applied to math [6], video game level design [11], and teaching the Thai alphabet [5]. This work also used test-input generation tools like Pex [58] and FShell [36], which systematically create test case suites with high code coverage, to generate problems for all possible traces (within certain bounds). However, this framework cannot be applied to non-procedural topics. For example, it is unclear how to analyze the execution trace of how a human understands natural language. We build on this work by proposing a general framework of problem decomposition and organization, which can be applied to non-procedural educational domains such as language learning. Within this framework, we can measure a student's ability and predict the student's performance on new problems.

Knowledge Assessment: IRT and KST

Item Response Theory (IRT) provides a framework for knowledge assessment [26, 27, 54]. IRT argues that the probability of a correct response to an item is a function of item parameters and individual ability [32]. Lord et al. proposed the 3PL model [46], which takes three item parameters into consideration: item difficulty, item discrimination, and the probability of guessing. A simpler model was proposed by Rasch [53], which stipulates that the probability of a correct response is determined only by the difference between the student's ability and the difficulty of the item. However, a common drawback of IRT models is that they measure student ability and item difficulty with unidimensional numeric scores [28], which does not reflect that students may find varying subsets of problems to be difficult depending on what they have mastered. We propose a novel way of measuring the difference between a student's ability and the difficulty of a problem that captures such variations. We validate this model with user data collected from an online knowledge assessment platform.

Knowledge Space Theory (KST) is a well-established perspective for studying the hierarchical structure of knowledge and a powerful tool for knowledge assessment [23, 24]. There are several tutoring systems based on KST, such as Alexs [28], RATH [35] and one for learning organic chemistry [57]. According to KST, a student's knowledge is represented as a knowledge state, the set of the problems that the student can solve. Problems are organized into a knowledge structure, which contains all possible knowledge states as well as the connections between these states [2, 40, 41]. Researchers have proposed approaches for building knowledge structures, such as querying experts [41] and Bayesian inference [21]. Due to the complexity of representing a knowledge state, Falmagne et al. introduced the idea of a "fringe" to characterize a student's knowledge, which can be calculated by an "entropy"-based approach [28]. We build on this work by proposing an automatic framework that can decompose a problem into its prerequisite basic skills and builds the hierarchical structure for a set of problems. Then, our framework can predict a student's performance on new problems by measuring the relationship of the student's ability to the problem within this structure.

PARTIAL ORDERINGS FOR NON-PROCEDURAL SKILLS

In order to select practice problems at an appropriate difficult level for each student, we need a hierarchical structure that encodes difficulty relationships between problems. One straightforward way to do this is to ask experts to specify these relationships. However, this becomes prohibitively difficult as the size of the problem set grows larger. Ideally, we would have automatic methods of problem organization.

In previous work [6, 12], researchers built partial ordering graphs for procedural tasks. This work used procedural execution traces to organize content and partial orderings over those traces to create hierarchical content structures. For example, one can identify at least four basic skills that may be required to solve an integer addition problem: one-digit addition without a carry (A), one-digit addition with a carry (B), writing a carry (C), and bringing down a final carry (D). As an example, problems can be decomposed into basic skills as follows:

Problem	2+3	15+18	93+15	298+865
Trace	A	ACB	AACD	ACBCBD

However, in some domains, such as language learning, the target knowledge cannot easily be modeled as a single procedure. To induce hierarchical knowledge structures in such domains, we need to generalize beyond procedural domains. To do this, our framework takes advantage of *compositionality*, the idea that problems can be broken down into smaller conceptual units. This is well-studied in some semi-procedural domains such as math and language learning [6, 60].

For instance, a Japanese sentence can be decomposed into *grammatical templates*, units of grammar that expert language instructors and linguists have identified as the most important grammatical knowledge, and that students actually study in language lessons. These templates have proved to be beneficial for text difficulty evaluation for language learners [60]. We found that the specific task of grammatically understanding a Japanese sentence can be decomposed into a (multi)set of grammatical templates. For example, here are three Japanese sentences and their grammatical templates:

S1:	私	の	先生		
	I	of	teacher		
	" my teacher "				
Templates:	(– の)				
S2:	私	の	先生	は	忙しい
	I	of	teacher	(topic)	busy
	" my teacher is busy "				
Templates:	(– の) (– は)				
S3:	私	の	先生	の	名前
	I	of	teacher	of	name
	" my teacher's name "				
Templates:	(– の) (– の)				

We can see that S1 has only one grammatical template: (– の). For a Japanese learner, S2 is harder than S1 since it has not only (– の), but also another template (– は). S3 repeats the same template (– の) twice hence is also harder than S1. These relationships cannot be captured by the partial ordering in [6] since it is unclear how to proceduralize the process of how a human understands these sentences. However, by considering multisets of concepts rather than execution traces, we can accommodate them as follows:

Definition 1a. A problem s can be decomposed into a series of concepts (basic skills) required by problem s. Since problems may require students to repeat some skills one or more times, we use a *multiset* of basic skills, indicated as $p(s)$, to characterize the difficulty of a problem.

Definition 1b. We say problem s_1 is *at least as hard as* s_2, indicated as $s_1 \geq s_2$, if and only if $p(s_1) \sqsupseteq p(s_2)$. This implies that if a student can solve s_1, he/she must be able to solve s_2 as well [6].

Here $\sqsupseteq$ denotes the superset relation between multisets. If $p(s_1) \sqsupseteq p(s_2)$, then for any concept c that is required by problem s_2 n times, c must also be required by s_1 at least n times. For example, AABC $\sqsupseteq$ ABC is true, while AABCC $\sqsupseteq$ ABBC is not true since ABBC has two "B"s while AABCC has only one.

Definition 1c. The strict partial order $s_1 > s_2$ is defined as $s_1 \geq s_2 \wedge s_2 \ngeq s_1$, which means s_1 is *(strictly) harder than* s_2.

Definition 1d. We say problem s_1 is *directly harder than* s_2, if and only if $s_1 > s_2$ and there is no other problem s_3 such that $s_1 > s_3 > s_2$.

Using Definition 1d, we build the hierarchical structure for a set of problems as follows:

Definition 2. We organize a set of problems $S = \{s_1, s_2, \cdots\}$ (we call S the *universal problem set*) as a *partial ordering graph* $G = \langle S, E \rangle$, where

$$E = \{(s_i, s_j) | s_i, s_j \in S \wedge s_j \text{ is directly harder than } s_i\} \quad (1)$$

Namely, there is an (directed) edge from problem s_i to s_j in the partial ordering graph if and only if s_j is directly harder than s_i. An example of a partial ordering graph is shown in Figure 1.

THE KNOWLEDGE BOUNDARY

Up until now, we have described how to build a partial ordering graph. In order to give practice problems that are at the appropriate difficulty level we need to assess students' knowledge. This is challenging because every student will understand a different subset of the problems. Therefore, we introduce the idea of the *knowledge boundary*, the "fringe" of knowledge that we use to characterize a student's ability in the partial ordering graph[1].

[1]Knowledge Space Theory defined the fringe of knowledge as a simpler representation of a student's knowledge state. However, we define the knowledge boundary in a partial ordering graph, which provides a way to measure how far away a problem is from what a student currently knows.

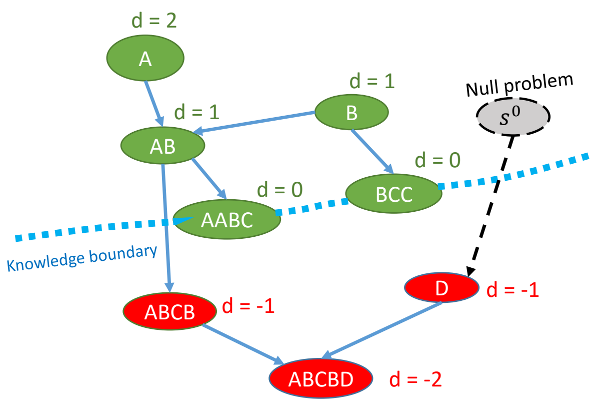
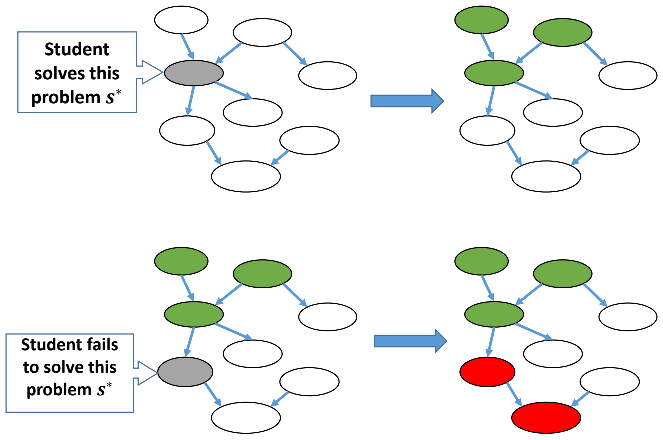


Figure 1. A Student's Knowledge boundary in the partial ordering graph. Uppercase letters represent concepts and the nodes with strings represent problems. The student can solve the green problems while cannot solve the red ones. The distances form each problem to the knowledge boundary are indicated as d. The null problem s^0 is an anxious problem that is added to help us compute the distances.



Student solves this problem s^*
Student fails to solve this problem s^*

Figure 2. Coloring the partial ordering graph based on the student's response. If the student solves s^*, all the nodes that are at most as hard as s^* (including s^* itself) will be colored 'solvable' (green); if the student fails to solve s^*, all the nodes that are at least as hard as s^* (including s^* it self) will be colored 'unsolvable' (red).

Definition 3. We measure a student's ability with the *knowledge boundary* K, which is defined as the hardest problems that the student can solve. Formally, if T is the set of the problems that the student can solve, then:

$$K = \{s | s \in T \wedge \nexists\, s' \in T \text{ such that } s' > s\} \qquad (2)$$

Consider the partial ordering graph in Figure 1 as an example. Assume a student can solve problems A, AB, B, AABC and BCC (the green nodes in the graph), and cannot solve ABCB, D or ABCBD (the red nodes in the graph). Then the knowledge boundary consists of only two problems: AABC and BCC, since there are no other "green" problems that are harder than AABC or BCC. The knowledge boundary does not include A, AB or B since there is a "green" problem AABC that is harder than all of them.

CALCULATING THE KNOWLEDGE BOUNDARY

We present a graph coloring algorithm of calculating a student's knowledge boundary in the partial ordering graph. This algorithm is based on two properties of the partial ordering: if a student can solve problem s, he must be able to solve any problem s' that is at most as hard as s ($s \geq s'$); if a student cannot solve problem s, he must not be able to solve any problem s' that is at least as hard as s ($s' \geq s$). For example, if a student can solve problem AB, he/she can also solve problems A and B; if a student cannot solve problem ABCB, then he/she cannot solve problem ABCBD either.

At the start of the algorithm, all the problems (nodes) in the partial ordering graph are uncolored. The algorithm asks the student whether he/she can solve some problem s^*. If the student can solve s^*, all the nodes that are at most as hard as s^* (including s^* itself) will be colored 'solvable'; if the student cannot solve s^*, all the nodes that are at least as hard as s^* (including s^* it self) will be colored 'unsolvable'. Figure 2 shows how this coloring process works.

The algorithm repeatedly selects an uncolored problem s^* from the partial ordering graph, asks the student to solve it, and then updates the coloring of the graph based on the response. This is a greedy algorithm designed to minimize the number of problems that must be given to the student. Formally, if n_s^+ denotes the number of the uncolored problems that are at most as hard as s, and n_s^- denotes the number of the uncolored problems that are at least as hard as s, then we can maximize the number of problems that can be colored based on the student's response by greedily selecting a problem s^* as follows:

$$s^* = \underset{s \text{ is uncolored}}{\arg\max} \; \min(n_s^+, n_s^-) \qquad (3)$$

PREDICTING PERFORMANCE ON SPECIFIC PROBLEMS

Distance to Knowledge Boundary

In order to recommend problems at appropriate difficulty levels to the students, we need to predict students' performance on problems. Existing IRT studies have proposed several popular models stipulating how student performance is related to student ability [46, 53]. However, they measure a student's ability and the difficulty of a problem using unidimensional numeric scores, which is incomprehensive [28].

Ideally, prediction of a student's performance would utilize multidimensional metrics to measure the distance between a problem and what a student already knows. The key technical challenge in the design of multidimensional metrics is that it is impossible to measure this distance without taking into account the hierarchical structure of the problem space. In our framework, we can measure this as the distance from the problem to the knowledge boundary in the partial ordering graph. We use signed numbers to distinguish which 'side' of the knowledge boundary a problem is on: problems 'inside' the boundary (which the student can solve) have positive distances while problems 'outside' the boundary (which the student cannot solve) have negative distances. Using this distance, we can leverage IRT models to predict a student's performance on new problems.

Here, we give the definition of this distance together with the examples in Figure 1. Assume we have the universal problem set S, and a student can solve a subset of problems T. For example, in Figure 1, $S=$\{A, AB, B, AABC, BCC, ABCB, D, ABCBD\}, $T=$\{A, AB, B, AABC, BCC\}, and the knowledge boundary $K=$\{AABC, BCC\}. We calculate the distance from any problem $s \in S$ to the knowledge boundary K, indicated as $d(s,K)$, following the steps below:

Step 1: Calculate distances for problems on the boundary
For any problem s such that $s \in K$, $d(s,K)=0$.

For example, the distance of problems AABC and BCC is 0.

Step 2: Calculate distances for problems inside the boundary
For any problem s such that $s \in T-K$, based on the definition of K, there must be one or more $s' \in K$ such that $s' \geq s$ (otherwise, s should be contained in K) , and

$$d(s,K)=\min_{s' \in K} dis(s,s') \tag{4}$$

where $dis(s,s')$ indicates the length of the shortest directed path from s to s' in the partial ordering graph. If there is no directed path from s to s', $dis(s,s')=\infty$. Note that if $s' \geq s$, then there must exist at least one directed path from s' to s.

For example, the distance of problem AB is 1, since problem AABC, which is on the knowlegde boundary, is directly harder than AB. Similar for problem B (BCC is direclty harder than B). The distance of problem A is 2, since the shortest directed path from A to any problem on the knowledge boundary (which is A→AB→AABC) has length 2.

Step 3: Calculate distances for problems outside the boundary
In a hierarchical knowledge structure, it follows intuitively that problems that are further away from the boundary will be more difficult to the student. Therefore, for any problem s such that $s \in S-T$, we define the distance $d(s,K)$ to be shortest directed path from any problem in T to s. Note that this distance also has a teaching interpretation: if easier problems should always be taught before harder problems [55], then this distance also measures the number of problems that need to be taught before teaching s.

Since there are some basic problems that have no problems easier than them, and thus have no incoming edges in the partial ordering graph, there is not always a path from a problem in T to s. For instance, there is no directed path from any problem in T to problem D in Figure 1. To resolve this, we add a *null problem* s^0, the pseudo problem with no prerequisite concepts, to T. For any problem $s \in S-T$, if there is no other problem s' such that s is directly harder than s' (namely, s has no incoming edges), we add an edge (s^0,s) to the partial ordering graph.

Now there is at least one directed path from the null problem s^0 or some problem in T to s. We can define the distance as:

$$d(s,K)=-\min_{s' \in T \cup \{s^0\}} dis(s',s) \tag{5}$$

Note that $d(s,K)$ is negative if and only if s is outside the knowledge boundary.

For example, the distance of problem ABCB is -1, since it is directly harder than the "green" problem AB. Problem D has no incoming edges in the partial ordering graph, hence we add an edge from the null problem s^0 to D, and the distance of problem D is -1 since the path s^0 →D has length 1. Lastly, the distance of problem ABCBD is -2. Actually, there are two shortest paths with length 2: AB→ABCB→ABCBD and s^0 →D→ABCBD.

The metric of distance is dependent on the density of the partial-ordering graph. This is inevitable since the measurement is based on the hierarchical structure of the problem space. We believe that for most well-defined problem spaces, it is a reasonable assumption that the partial ordering graph will be sufficiently dense. We will demonstrate this metric works well for a well-built Japanese language learning corpus in the next section.

In the later part of this paper, we will use d to denote $d(s,K)$ for convenience.

Adapted Rasch model

In this section, we describe how we can adapt existing unidimensional IRT models to build a multidimensional metric that leverages the partial ordering graph and the distance d calculated in the previous section in order to predict student performance. One of the most famous models of IRT, the Rasch model [53], stipulates that a student's performance P is a function of the difference between the student's ability θ and the problem's difficulty b:

$$P(\theta,b)=\frac{e^{\theta-b}}{1+e^{\theta-b}} \tag{6}$$

In the Rasch model, $\theta-b$ measures the difference between the student's ability and the difficulty of the problem. In our framework, we use d to measure this, hence we want to replace $\theta-b$ with d. It is also common to add a *discrimination* parameter a, which represents the degree to which the task discriminates between students [19]. In addition, we have found in practice that we need to add an additional parameter c, which measures how 'comfortable' the students feel with the problems on the knowledge boundary (any problem s such that $s \in K$). Ideally, students should be able to solve the problems on the boundary. However, in reality, students may still experience some difficulty with these problems. We can thus replace $\theta-b$ with $ad+c$.

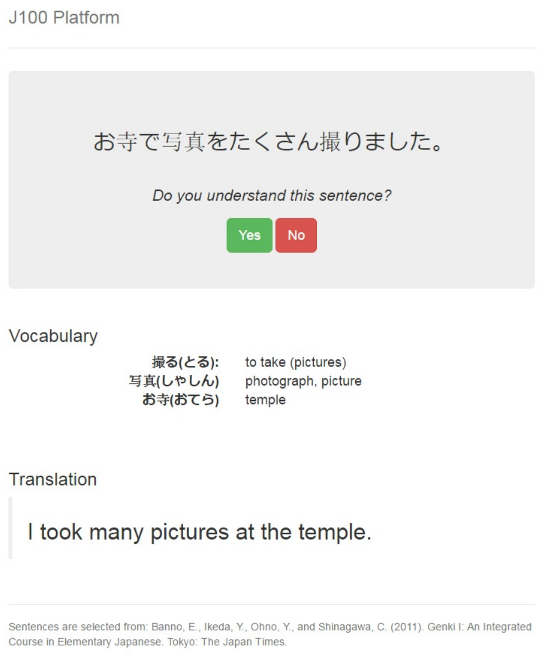
This brings us to an adapted Rasch model, which stipulates how student performance P is related to the distance d:

$$P(d)=\frac{e^{ad+c}}{1+e^{ad+c}} \tag{7}$$

In the next section, we will demonstrate that this adapted model nicely fits the data collected from our knowledge assessment platform.

EVALUATION OF KNOWLEDGE ASSESSMENT

In this section, we will the evaluate our calculation of the knowledge boundary and the distance presented in the previous section by applying it to a Japanese language learning domain.


J100 Platform

お寺で写真をたくさん撮りました。

Do you understand this sentence?

Vocabulary

撮る(とる):	to take (pictures)
写真(しゃしん)	photograph, picture
お寺(おてら)	temple

Translation

I took many pictures at the temple.

Sentences are selected from: Banno, E., Ikeda, Y., Ohno, Y., and Shinagawa, C. (2011). Genki I: An Integrated Course in Elementary Japanese. Tokyo: The Japan Times.

Figure 3. Screenshot of J100 Platform (Assessment Stage)

J100: A Language Assessment Platform

J100 is a language assessment platform that evaluates how well a user understands the grammatical knowledge in *Genki I* [9] (JLPT level N5[2]). It is designed for Japanese beginners who have learned Japanese for less than 1 year. As the platform starts, users view 15-18 Japanese sentences as well as corresponding vocabulary explanations and sentence translations[3]. The J100 platform will ask users to judge how well they understand each sentence. All of the test sentences were collected from *Genki I*. Figure 3 shows a screenshot of the J100 platform.

J100 has 2 stages: *assessment* and *evaluation*. In the assessment stage, users view 10 sentences and respond whether they understand each sentence ('Yes' or 'No'). Using the graph-coloring algorithm, we calculate a user's knowledge boundary based on his or her responses in this stage. In the evaluation stage, users view 5-8 additional sentences and are asked to indicate how well they understand each sentence ('Yes', 'Almost', 'Somewhat', 'Little', or 'No'). Sentences used in the assessment stage are not repeated in the evaluation stage. The responses in the evaluation stage are used to validate the knowledge boundary calculated in the assessment stage.

Reddit Deployment

We recruited users through the Japanese Learning sub-Reddit[4]. The deployment was very successful: 847 users finished the J100 'test', and our post received an up-vote/down-vote score of 145 (for comparison, the other posts on the same page have an average score of 12). We received about 50 comments from J100 users, and most of the comments are supportive *"This is ideal. Especially good for learners like myself who tend to forget bits and details of older lessons, and just good in general for testing your progress."* Many users even expressed a future interest in our platform and requested that we make it work for higher levels (N4-N1): *"My friends love this. If you could make some intuitive tests for N4-N1 I'd pay money if I could take tests that could also adapt to my level. Making me learn little by little what I need improving on."*

[2]Japanese Language Proficiency Test (JLPT) has 5 levels: from N1 (advanced) to N5 (beginner). You can see a summary of JLPT levels here: http://www.jlpt.jp/e/about/levelsummary.html

[3]The vocabulary explanations are provided to users since J100 is focusing on grammatical knowledge only, while users are not recommended to read sentence translations unless they are not sure whether their understanding of the sentence is correct.

[4]www.reddit.com/r/LearnJapanese

Validating the Knowledge Boundary and Distance Metric

In order to validate our calculation of the knowledge boundary and the distance metric, we will demonstrate that the user data collected from J100 nicely fits the adapted Rasch Model. We calculate a user's knowledge boundary K based on his/her responses to the problems in the assessment stage, and for each problem s responded in the evaluation stage, we measure the distance from the problem s to the knowledge boundary K. The user responses in the evaluation stage can be regarded as users' self-estimation of their performance, and we score the five possible responses uniformly from 1 to 0:

Response	Yes	Almost	Somewhat	Little	No
Score	1	0.75	0.5	0.25	0

We would like to test if the relationship between the distance and user response matches the adapted Rasch model (Equation 7). If it it does match, then this result will verify our calculation of the knowledge boundary as an effective measurement of a student's knowledge. Additionally, we can utilize the distance from a problem to the student's knowledge boundary to predict the student's performance on that problem.

We calculate each user's knowledge boundary after the assessment stage, and calculate the distance to each problem given in the evaluation stage to that knowledge boundary. Then we group these problems by these distances and calculate the the average and standard deviation of the user responses when given problems with the same distance (Table 1).

We use non-linear regression to estimate the parameters a and c, and show the results in Figure 4. The results suggest that the adapted Rasch model fits our user data fairly well ($R^2 = 0.992$). There are some imperfections in the top-right corner of the figure, where users tended to indicate less confidence in their understanding of the sentence than would have been expected. This may be because users find it hard to judge whether they 'understand' or 'almost understand' a sentence, and may have slightly underestimated their ability. It could also result from the gap between the scores of 'Yes' (1) and 'Almost' (0.75) - if a user wanted to indicate something in between, such as 0.9, no such response was available.

There is another interesting phenomenon: the standard deviation of the user responses (in Table 1) is maximized where the distance is at some point between -2 and -3, and the slope of the curve in Figure 4 is also maximized where the distance is at some similar point between -2 and -3. According to IRT, the

Distance	User Response	
	Average	StdDev.
-4	0.125	0.11547
-3	0.279412	0.329654
-2	0.638037	0.40631
-1	0.861589	0.266999
0	0.913887	0.198634
1	0.940882	0.171181
2	0.959838	0.144577
3	0.983871	0.11543

Table 1. The average and standard deviation of J100 user responses, grouped by the calculated distance from the knowledge boundary. Note that the standard deviation of the user responses is maximized where the distance is at some point between -2 and -3, indicating that users are most unsure of their ability at this distance.

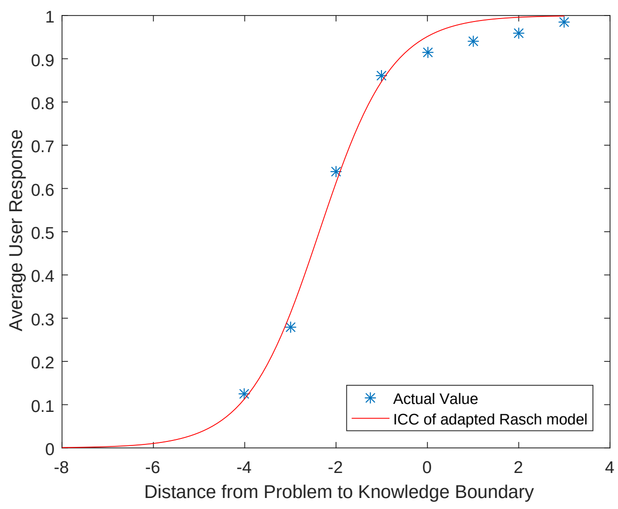


Figure 4. The adapted Rasch model nicely fits the J100 user data. The blue data points show the average user response to problems with the same distance from the knowledge boundary, and the red curve is the *Item Characteristic Curve* (ICC) calculated based on the adapted Rasch model.

problems that are at the difficulty level that corresponds to the maximum slope of the curve are the most discriminative, and the J100 data shows that users are most unsure of their ability at this distance. This makes sense and is further evidence that the adapted Rasch model fits the J100 user data well.

In conclusion, these results validate that our framework can efficiently measure students' ability with the knowledge boundary and predict student performance on specific problems. Since our formalism is inspired by both Item Response Theory (IRT) and Knowledge Space Theory (KST), the idea of the knowledge boundary and the distance may contribute to the unification of these two theories.

PROGRESSION ANALYSIS

Good progressions enhance students' engagement of the curriculum [43,56], and our work aims to discover general characteristics of good progressions. In this section, we will leverage our framework to analyze educational progressions, and introduce two significant features of a good progression: composition and pace. We study existing curricula and demonstrate some striking similarities in expert-designed progressions.

Composition: The Balance of Learning and Review

When designing an educational progression, a key consideration is how much time one should spend reinforcing previously-introduced knowledge as opposed to introducing new knowledge. Another consideration is whether the progression should grow in complexity by combining concepts together in new ways. Ideally, we would be able txo define a progression parameter space that can concisely capture these important aspects.

In our framework, for any problem s that the student is about to learn, we classify s, with respect to the current knowledge boundary K, as:

reinforcement, if $\exists s' \in K$ s.t. $s' \geq s$;

recombination, if $\nexists s' \in K$ s.t. $s' \geq s$, and for any concept c that is required by s, $\exists s' \in K$ s.t. s' also requires concept c.

introduction, if there exists some concept c s.t. c is required by s but is not required by any problem in K.

In other words, the next problem in the progression will be classified as reinforcement if the student has learned some problem that is at least as hard as it, as introduction if it requires any new concepts (basic skills) that the student has never learned, and as recombination if it combines together previously learned concepts in a new way.

For example, if the current knowledge boundary is:

$K = \{A, BCD, AFFD, BBECC\}$

The following problems will be classified as:

reinforcement: $A, BD, AFD, BBEC$

recombination: $AB, BFE, BBEECC, AFDD$

intruduction: $G, BHD, BBEGC, GH$

We can think of reinforcement as the review of learned knowledge, introduction as pure learning, and recombination as a mixture of both. The designer's strategy of balancing learning and review in an educational progression is revealed by the proportions of these three types, which we refer to as *progression composition*.

Pace: The Growth Rate of Knowledge

We would like to know if the difficulty of lessons (chapters, units, etc.) in a progression grows at a consistent speed such that students can learn smoothly. Instead of numerically measuring the difficulty of each lesson, we calculate the *knowledge size*, the total number of problems that have been introduced up until each lesson. This can be measured as the number of the problems in the universal problem set[5] S that are classified as reinforcement by the student's knowledge boundary K after mastering all the knowledge from the start to the current lesson. The knowledge size $|K|$ is therefore the number of such problems:

$$|K| = |\{s \in S \wedge \exists s' \in K \text{ s.t. } s' \geq s\}| \tag{8}$$

[5]The universal problem set S contains all the practice problems we have for an educational task.

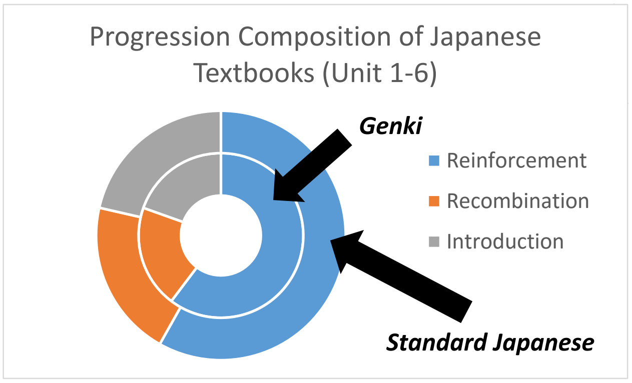


Figure 5. Progression composition of two Japanese textbooks: *Standard Japanese* and *Genki*. The proportions are fairly similar. Note that it usually takes a student 2 semesters to learn units 1-6.

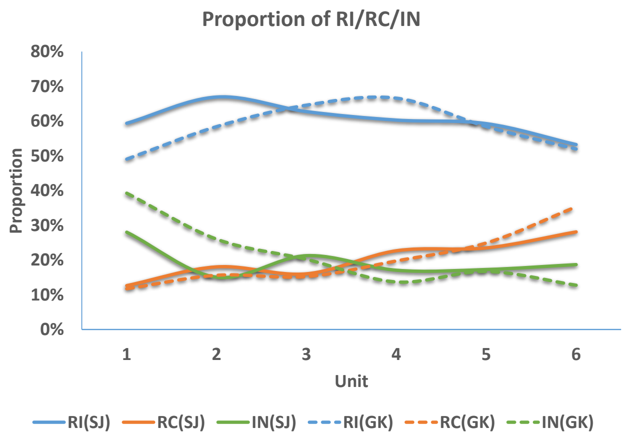


Figure 6. Proportion of reinforcement(RI), recombination(RC) and Introduction(IN) in two series of Japanese textbooks: *Standard Japanese* (SJ) and *Genki*(GK). Here x-axis represents the learning progress: unit 1-6, and y-axis demonstrates the proportions.

The *pace* is the ratio between knowledge growth (change of knowledge size $\Delta|K|$) and time. Assuming that the number of problems that students learn is directly proportional to the amount of expended time, we can use the length of the lesson $|L|$, which measures the number of practice problems in it, to represent time. Hence, the pace of lesson i can be measured as:

$$Pace_i = \frac{|K_i| - |K_{i-1}|}{|L_i|} \quad (9)$$

where $|K_i|$ and $|K_{i-1}|$ indicate the knowledge sizes after lesson i and lesson $i-1$.

For example, assume the knowledge size after lesson 5 is 110, and the knowledge size after lesson 6 is 134. If lesson 6 has 8 problems, then the pace of lesson 6 is (134-110)/8 = 3.

EVALUATION OF PROGRESSION ANALYSIS

Studies on Textbooks

To validate our work, we study two popular Japanese textbooks: *Standard Japanese* [51] and *Genki* [9]. Note that in language learning, 'problems' are the practice sentences that are introduced throughout the curriculum. Our goal is to find out if the progressions of both textbooks have similar pace and composition.

Both Textbook Progressions Have Similar Composition

Figure 5 shows the progression compositions of the first 6 units (which usually takes a student two semesters to learn) in these two Japanese textbooks. Clearly, both textbook progressions have a similar composition, possibly revealing that the experts who designed these two books arrived at similar conclusions.

To analyze how the progression composition changes over time, we calculate the proportions of reinforcement, recombination and introduction of each unit of both textbooks. For the proportions of unit i, we generate the knowledge boundary K_{i-1} based on unit 1 to unit $i-1$, and classify all sentences in unit i based on K_{i-1}.

Figure 6 shows how the composition of these two textbook progressions changes over time. Clearly, there are some striking similarities between the curves of reinforcement, recombination and introduction: for both series of books, the proportion of reinforcement increases to 70% then drops to 50%, the proportion of recombination increases from 10% to 30%-35%, and the proportion of introduction decreases from 30%-40% to 15%-20%. This validates the usefulness of characterizing progression composition, as there may be general principles of balancing learning and review that can be learned from existing textbooks.

Both Textbook Progressions Have a Similar, Steady Pace

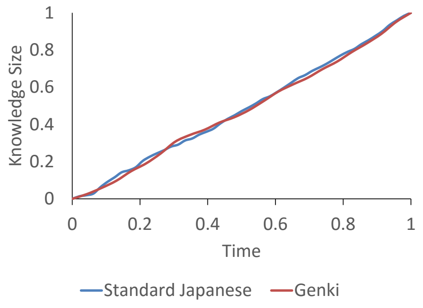
Figure 7a shows the pace of both textbooks. Clearly, they both introduce the same amount of knowledge over the same period of time, which demonstrates they use a steady pace. This fits our intuition that learning is most comfortable with a steady pace.

To test if other reasonable progressions also have this property, we generate two alternate progressions by shuffling the sentences. The first alternate progression is generated randomly, and the second follows a topological sort. The topological sort ensures that any sentence s is put before all sentences that are harder than s. We compare the textbook progressions with the progressions generated by the random algorithm and topological sort (Figure 7b and Figure 7c). The random progression is too difficult at the beginning because hard and easy sentences are equally likely to be selected. This may cause students to become discouraged by hard sentences at the beginning. On the other hand, the topological sort is too easy at the beginning because all of the easy sentences are put first. This results in an increase of difficulty that is initially too slow. As can be seen from our analyses of these two textbooks, steady pacing may be an important characteristic of a good progression.

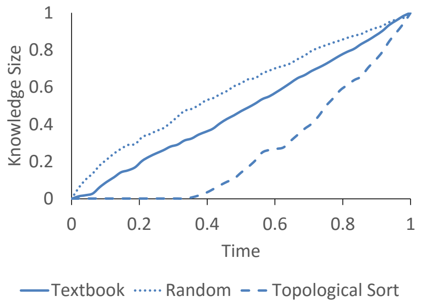
Studies on Online Language Learning Tools

We also analyze the progressions used by two online Spanish learning tools: Duolingo[6] and Language Zen[7]. We observed that some of the interesting differences in these two language

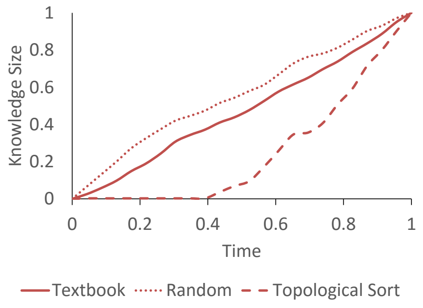
[6]https://www.duolingo.com/course/es/en/Learn-Spanish-Online
[7]www.languagezen.com



(a) Paces of Textbook Progressions: *Standard Japanese* and *Genki*.



(b) Standard Japanese Texts: Paces of textbook progression, and the progressions generated by the random algorithm and topological sort.



(c) Genki Texts: Paces of textbook progression, and the progressions generated by the random algorithm and topological sort.

Figure 7. Pace of various progressions. Here, the x-axis represents the normalized learning time (namely, the normalized number of sentences learned), and the y-axis demonstrates the normalized knowledge size. (a) shows that both textbooks introduce the same amount of knowledge over the same period of time, which indicates that they use a steady pace. (b) and (c) compare the textbook progression to other baseline progressions. When compared with the textbook progression, random progressions are too difficult at the beginning, and progressions generated by a topological sort are too easy at the beginning.

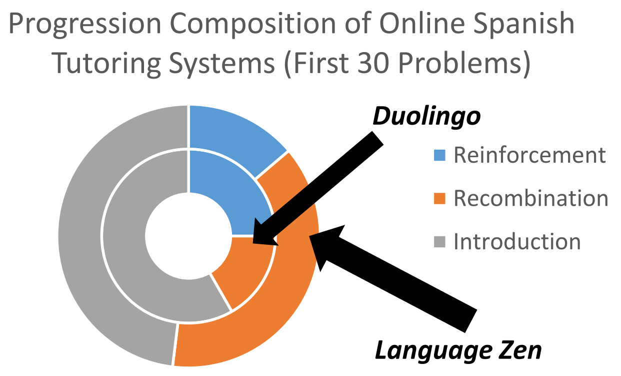


Figure 8. Progression composition of two online Spanish learning tools: Duolingo and Language Zen. Compared with Duolingo, Language Zen has less reinforcement and less introduction, but much more recombination. Note that it usually takes a student a couple of days to learn 30 problems.

learning progressions resulted from how they introduce vocabulary. Therefore, instead of using grammatical templates for Spanish, we treated each unique word as its own concept, and calculated the knowledge boundary and progression parameters using vocabulary.

The progression compositions of the first 30 problems (which usually take a student a couple of days to learn) introduced in these two tools are shown in Figure 8. Compared with Duolingo, Language Zen has less reinforcement and less introduction, but much more recombination. It has been claimed[8] that Language Zen is 1.5 times faster than Duolingo. We speculate that the higher proportion of recombination may contribute to this difference.

One may also notice that Figure 5 has a higher proportion of reinforcement and a lower proportion of introduction than Figure 8. There are three factors we believe that could contribute to these differences. One factor is the learning method. Textbooks are typically used in classroom learning and are thus not necessarily intended to be the only learning material that students are using. They are typically combined with lectures and assignments. However, online language learning tools such as Duolingo and Language Zen are more self-contained. Therefore, these textbooks may have a higher proportion of reinforcement (review) than these online tutoring systems.

The second factor is the difference of the learning period. The texts in Figure 5 are usually taught over two semesters, while Figure 8 only shows the sentences that are taught in the first couple of days of the online learning tools. The differences in progression composition in the two figures imply that these expert-designed progressions tend to focus more on introduction than reinforcement at the very beginning of the progression. This is also consistent with Figure 6, where the proportion of introduction is the highest at the beginning then drops afterwards.

The third factor is that Figure 5 examines grammar and Figure 8 examines vocabulary, which could imply that grammar needs more reinforcement than vocabulary.

AUTOMATIC PROGRESSION SYNTHESIS

We have discussed two features of an educational progression: composition and pace. In this section, we will demonstrate that educational progressions can be automatically synthesized according to specific composition and pace parameters. To be more precise, a progression can be characterized as three numeric parameters: overall pace (*pace*), proportion of reinforcement (*ri*), and proportion of introduction (*in*). The proportion of recombination (*rc*) is redundant since $ri + rc + in = 1$.

We use a greedy algorithm to synthesize progressions. The algorithm starts with an empty progression, repeatedly selects the next problem that minimizes the following error function

[8]http://elmodenafrontline.com/8825/feature/language-zen-the-new-and-coming/

Progression Name	Desired Progression Parameters	Synthesized Progression
Genki Simulation	pace = 0.5, ri = 0.6, in = 0.2	DG B DG DG DG BG BG B BD B
Reinforcement-biased	pace=0.3, ri = 0.7, in = 0.2	C C C C C A C A AC A
Recombination-biased	pace =1.2, ri = 0.2, in = 0.3	G G AB BG AG BG EF BE AE BF
Introduction-biased	pace = 1.6, ri = 0.2, in = 0.5	EF EF AB BF ABC AF CDE AF BCD DG

Table 2. Synthesized progressions with desired parameters. The '*Genki* Simulation' progression is synthesized with parameters extracted from the textbook. The other three progressions are generated with specific parameter settings. This demonstrates the potential of our progression synthesis framework to map characteristics of one progression onto another, and to create progressions that are tailored to the needs of an individual student.

and appends it to the progression:

$$Error = (pace - pace^*)^2 + (ri - ri^*)^2 + (in - in^*)^2 \quad (10)$$

where $pace, ri, in$ are the actual progression characteristics, and $pace^*, ri^*, in^*$ are the desired progression parameters. Note that there may be multiple available problems with minimal error. In that case, our algorithm randomly selects one from them as the next problem.

This greedy algorithm does not always generate progressions with the exact desired parameters. However, it runs very fast. For a universal problem set of 25 problems, it can synthesize over 100 progressions (with 10 problems each) in one second. Therefore, we can run this greedy algorithm many times in order to synthesize progressions with desired characteristics.

Table 2 shows four examples of synthesized progressions. We extracted the following parameters from the *Genki* progression: $pace = 0.465, ri = 0.582, in = 0.214$. We then used these parameters (approximately) to synthesize the '*Genki* Simulation' progression. We also generated three other progressions with tailored parameter settings, which are biased towards reinforcement, recombination and introduction. Theses results demonstrate that we can not only utilize the principles of expert progression design and synthesize progressions with parameters that are good for most students, but also tailor progressions for students with specific preferences.

CONCLUSIONS AND FUTURE WORK

Learnability is important for many aspects of HCI: user interfaces, learning tools, video games, and software usability. In this paper, we defined a framework that utilizes automatic problem decomposition and partial ordering graph construction to facilitate multiple workflows related to education: assessment of a student's knowledge, identifying and describing the learning strategies of a curriculum, and designing new learning progressions. We evaluated this framework in multiple ways: through an online deployment of a Japanese-language assessment tool, automatic extraction of pacing parameters from existing curricula, and assembly of problems into new progressions that adhere to desired pacing characteristics.

We hope to utilize our framework for several educational domains, especially computer-assisted language education. For example, we plan to leverage our framework to recommend appropriate reading materials to second language learners and improve educational games for language learning [17, 18]. In the future, we will apply our framework not only to grammatical knowledge, but also other aspects of language learning, such as vocabulary and semantics.

We hope that these ideas will enable a science of progression analysis, in which sequencing and pacing parameters are extracted from progressions across a wide variety of topics to identify the best principles. Furthermore, we hope to extend this framework so that it can automatically build adaptive learning systems that are capable of both rapid initial assessment and progression tailoring.

REFERENCES

1. Sami Abuhamdeh and Mihaly Csikszentmihalyi. 2012. The importance of challenge for the enjoyment of intrinsically motivated, goal-directed activities. *Personality and Social Psychology Bulletin* 38, 3 (2012), 317–330.

2. Dietrich Albert and Theo Held. 1994. Establishing knowledge spaces by systematical problem construction. In *Knowledge structures*. Springer, 81–115.

3. Vincent Aleven, Bruce M McLaren, Jonathan Sewall, and Kenneth R Koedinger. 2006. The cognitive tutor authoring tools (CTAT): preliminary evaluation of efficiency gains. In *International Conference on Intelligent Tutoring Systems*. Springer, 61–70.

4. Vincent Aleven, Bruce M Mclaren, Jonathan Sewall, and Kenneth R Koedinger. 2009. A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education* 19, 2 (2009), 105–154.

5. Erik Andersen. 2014. *Automatic Scaffolding for Procedural Learning*. Ph.D. Dissertation.

6. Erik Andersen, Sumit Gulwani, and Zoran Popovic. 2013. A trace-based framework for analyzing and synthesizing educational progressions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 773–782.

7. John R Anderson, Albert T Corbett, Kenneth R Koedinger, and Ray Pelletier. 1995. Cognitive tutors: Lessons learned. *The journal of the learning sciences* 4, 2 (1995), 167–207.

8. John W Atkinson, Jarvis R Bastian, Robert W Earl, and George H Litwin. 1960. The achievement motive, goal setting, and probability preferences. *The Journal of Abnormal and Social Psychology* 60, 1 (1960), 27.

9. Eri Banno, Yoko Ikeda, and Yutaka Ohno. 2011. *GENKI: An Integrated Course in Elementary Japanese*. Japan Times and Tsai Fong Books.

10. Joseph E Beck, Kai-min Chang, Jack Mostow, and Albert Corbett. 2008. Does help help? Introducing the Bayesian Evaluation and Assessment methodology. In *International Conference on Intelligent Tutoring Systems*. Springer, 383–394.

11. Eric Butler, Erik Andersen, Adam M Smith, Sumit Gulwani, Zoran Popovic, and WA Redmond. 2015. Automatic Game Progression Design through Analysis of Solution Features. In *Proc. of the SIGCHI Conf. on Human Factors in Computing (CHI 2015)*.

12. Eric Butler, Adam M Smith, Yun-En Liu, and Zoran Popovic. 2013. A mixed-initiative tool for designing level progressions in games. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 377–386.

13. Herlinda Cancino, Ellen Rosansky, and John Schumann. 1978. The acquisition of English negatives and interrogatives by native Spanish speakers. *Second language acquisition: A book of readings* (1978), 207–230.

14. Albert T Corbett and John R Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction* 4, 4 (1994), 253–278.

15. Mihaly Csikszentmihalyi. 1990. *Flow: The Psychology of Optimal Experience*. Harper & Row Publishers, Inc., New York, NY, USA. 49 pages.

16. Mihaly Csikszentmihalyi. 1991. *Flow: The psychology of optimal experience*. Vol. 41. HarperPerennial New York.

17. Gabriel Culbertson, Erik Andersen, Walker White, Daniel Zhang, and Malte Jung. 2016a. Crystallize: An Immersive, Collaborative Game for Second Language Learning. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, 636–647.

18. Gabriel Culbertson, Shiyu Wang, Malte Jung, and Erik Andersen. 2016b. Social Situational Language Learning through an Online 3D Game. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 957–968.

19. Rafael Jaime De Ayala. 2013. *The theory and practice of item response theory*. Guilford Publications.

20. Edward L Deci and Richard M Ryan. 2000. The “what” and “why” of goal pursuits: Human needs and the self-determination of behavior. *Psychological inquiry* 11, 4 (2000), 227–268.

21. Michel C Desmarais, Peyman Meshkinfam, and Michel Gagnon. 2006. Learned student models with item to item knowledge structures. *User Modeling and User-Adapted Interaction* 16, 5 (2006), 403–434.

22. Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O’Hara, and Dan Dixon. 2011. Gamification. using game-design elements in non-gaming contexts. In *CHI’11 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2425–2428.

23. Jean-Paul Doignon and Jean-Claude Falmagne. 1985. Spaces for the assessment of knowledge. *International journal of man-machine studies* 23, 2 (1985), 175–196.

24. Jean-Paul Doignon and Jean-Claude Falmagne. 2012. *Knowledge spaces*. Springer Science & Business Media.

25. Tao Dong, Mira Dontcheva, Diana Joseph, Karrie Karahalios, Mark Newman, and Mark Ackerman. 2012. Discovery-based games for learning software. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2083–2086.

26. Fritz Drasgow and Charles L Hulin. 1990. Item response theory. (1990).

27. Susan E Embretson and Steven P Reise. 2013. *Item response theory*. Psychology Press.

28. Jean-Claude Falmagne, Eric Cosyn, Jean-Paul Doignon, and Nicolas Thiéry. 2006. The assessment of knowledge, in theory and in practice. In *Formal concept analysis*. Springer, 61–79.

29. Susan Gass. 1982. From Theory to Practice. (1982).

30. Jennifer M Goldschneider and Robert M DeKeyser. 2001. Explaining the “Natural Order of L2 Morpheme Acquisition” in English: A Meta-analysis of multiple determinants. *Language learning* 51, 1 (2001), 1–50.

31. Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. 2009. Generating photo manipulation tutorials by demonstration. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 66.

32. Ronald K Hambleton. 1991. *Fundamentals of item response theory*. Vol. 2. Sage publications.

33. Erik Harpstead, Brad A Myers, and Vincent Aleven. 2013. In search of learning: facilitating data analysis in educational games. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 79–88.

34. Susan Harter. 1978. Effectance motivation reconsidered. Toward a developmental model. *Human development* 21, 1 (1978), 34–64.

35. Cord Hockemeyer, Theo Held, and Dietrich Albert. 1997. RATH-A relational adaptive tutoring hypertext WWW-environment based on knowledge space theory. (1997).

36. Andreas Holzer, Christian Schallhart, Michael Tautschnig, and Helmut Veith. 2008. Fshell: Systematic test case generation for dynamic analysis and measurement. In *International Conference on Computer Aided Verification*. Springer, 209–213.

37. Katherine Isbister, Mary Flanagan, and Chelsea Hash. 2010. Designing games for learning: insights from conversations with designers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2041–2044.

38. Edward L Keenan and Bernard Comrie. 1977. Noun phrase accessibility and universal grammar. *Linguistic inquiry* 8, 1 (1977), 63–99.

39. Kenneth R Koedinger, John R Anderson, William H Hadley, and Mary A Mark. 1997. Intelligent tutoring goes to school in the big city. (1997).

40. Mathieu Koppen. 1993. Extracting human expertise for constructing knowledge spaces: An algorithm. *Journal of mathematical psychology* 37, 1 (1993), 1–20.

41. Mathieu Koppen and Jean-Paul Doignon. 1990. How to build a knowledge space by querying an expert. *Journal of Mathematical Psychology* 34, 3 (1990), 311–331.

42. Stephen D Krashen. 1985. *The input hypothesis: Issues and implications*. Addison-Wesley Longman Ltd.

43. Nan Li, William W Cohen, and Kenneth R Koedinger. 2012. Problem order implications for learning transfer. In *Intelligent Tutoring Systems*. Springer, 185–194.

44. Conor Linehan, George Bellord, Ben Kirman, Zachary H Morford, and Bryan Roche. 2014. Learning curves: analysing pace and challenge in four successful puzzle games. In *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play*. ACM, 181–190.

45. Derek Lomas, Kishan Patel, Jodi L Forlizzi, and Kenneth R Koedinger. 2013. Optimizing challenge in an educational game using large-scale design experiments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 89–98.

46. Frederic M Lord. 1980. *Applications of item response theory to practical testing problems*. Routledge.

47. Thomas W Malone. 1980. What makes things fun to learn? Heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*. ACM, 162–169.

48. Jürgen Meisel. 1987. Reference to past events and actions in the development of natural second language acquisition. *First and second language acquisition processes* (1987), 206–224.

49. Zachary Pardos and Neil Heffernan. 2010. Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence of the Expectation Maximization algorithm. In *Educational Data Mining 2010*.

50. Zachary A Pardos and Neil T Heffernan. 2011. KT-IDEM: Introducing item difficulty to the knowledge tracing model. In *International Conference on User Modeling, Adaptation, and Personalization*. Springer, 243–254.

51. People's Education Press. 2013. *Standard Japanese of China-Japan Exchanges for Beginners*. Mitsumura Tosho Publishing Co.Ltd.

52. Manfred Pienemann, Malcolm Johnston, and Geoff Brindley. 1988. Constructing an acquisition-based procedure for second language assessment. *Studies in second language acquisition* 10, 02 (1988), 217–243.

53. Georg Rasch. 1993. *Probabilistic models for some intelligence and attainment tests*. ERIC.

54. Mark Reckase. 2009. *Multidimensional item response theory*. Vol. 150. Springer.

55. C Reigeluth and R Stein. 1983. Elaboration theory. *Instructional-design theories and models: An overview of their current status* (1983), 335–381.

56. Charles M Reigeluth. 2013. *Instructional design theories and models: An overview of their current status*. Routledge.

57. Mare Taagepera and S Noori. 2000. Mapping students' thinking patterns in learning organic chemistry by the use of knowledge space theory. *J. Chem. Educ* 77, 9 (2000), 1224.

58. Nikolai Tillmann and Jonathan De Halleux. 2008. Pex–white box test generation for. net. In *International conference on tests and proofs*. Springer, 134–153.

59. Lev Semenovich Vygotsky. 1980. *Mind in society: The development of higher psychological processes*. Harvard university press.

60. Shuhan Wang and Erik Andersen. 2016. Grammatical Templates: Improving Text Difficulty Evaluation for Language Learners. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*. 1692–1702.

61. Joanna White. 1998. Getting the learners' attention: A typographical input enhancement study. *Focus on form in classroom second language acquisition* (1998), 85–113.

62. Henning Wode. 1981. LANGUAGE-ACQUISITIONAL UNIVERSALS: A UNIFIED VIEW OF LANGUAGE ACQUISITION. *Annals of the New York Academy of Sciences* 379, 1 (1981), 218–234.

63. Ruth Wylie, Kenneth Koedinger, and Teruko Mitamura. 2010. Analogies, explanations, and practice: examining how task types affect second language grammar learning. In *International Conference on Intelligent Tutoring Systems*. Springer, 214–223.

64. Yanbo Xu and Jack Mostow. 2010. Using logistic regression to trace multiple sub-skills in a dynamic bayes net. In *Educational Data Mining 2011*.

65. Helen Zhao, Kenneth Koedinger, and John Kowalski. 2013. Knowledge tracing and cue contrast: Second language English grammar instruction. In *Proceedings of the 35th Annual Conference of the Cognitive Science Society*.

66. Helmut Zobl. 1984. The wave model of linguistic change and the naturalness of interlanguage. *Studies in Second Language Acquisition* 6, 02 (1984), 160–185.