7. Domain modeling for AIED systems with connections to modeling student knowledge: a review

Vincent Aleven, Jonathan Rowe, Yun Huang and Antonija Mitrovic

INTRODUCTION

A central component of many AIED systems is a "domain model," that is, a representation of knowledge of the domain of instruction. The system uses the model in many ways to provide instruction that adapts to learners. Not all AIED systems have an elaborate domain model, but in those that do, the domain model is central to the system's functioning. In fact, domain models fulfill so many important functions within AIED systems that entire classes of AIED systems are defined in terms of the types of domain model they use (such as model-tracing tutors, constraint-based tutors, example-tracing tutors, and issue-based approaches to building tutoring systems). Across AIED projects, systems, and paradigms, the types of domain models used span the gamut of AI representations. AIED systems use their domain models for many different purposes, chief among them assessing student work, which is foundational for other functionality.

This chapter reviews major approaches to domain modeling used in AIED systems and briefly touches on the corresponding student models and the way they are used to track an individual student's knowledge growth. (We do not discuss student models that target other aspects, such as affect, motivation, self-regulation, or metacognition.) We discuss, in turn: rule-based models, constraint-based models, Bayesian networks, machine-learned models, text-based models, generalized examples, and knowledge spaces. These types of models have been studied extensively in AIED research and have been the foundation for many AIED systems that have been proven to be effective in enhancing student learning or other aspects of the student experience. A number of these approaches are now used in AIED systems that are used on a wide scale in educational practice. The chapter discusses how these approaches support key aspects of an AIED system's behavior and enable the system to adapt aspects of its instruction to individual student variables. We also highlight challenges that occur when applying the different approaches. We look at the use of machine learning and data-driven methods to create or refine domain models, so they better account for learning data and support more effective adaptive instruction. As well, we make note of connections between a system's domain model and other key components, including the system's student model. We base this discussion on traditional views of intelligent tutoring systems (ITSs), which divide the system's architecture into four main components: a domain model, a student model, a pedagogical model, and a problem-solving environment. We focus on systems that support *individual learning*. Other types of AIED systems are covered in other chapters.

What Do We Mean by a Domain Model?

The domain model of an AIED system captures knowledge in the given task domain, including concepts, skills, strategies, tactics, or constraints. In many cases, it captures the knowledge that the system aims to help students learn. A domain model in an AIED system normally contains the ideal knowledge that experts have. It may optionally contain representations of incorrect knowledge that novices in the domain tend to exhibit, such as bugs, mal-rules, and misconceptions. In addition, a domain model may capture prerequisite relations and other relations between the knowledge components represented in the model, as well as relations between knowledge components and practice problems or steps of practice problems, also referred to as "items." This definition is grounded in older definitions of the domain model but is broader (see Burns & Capps, 1988; Holmes, Bialik, & Fadel, 2019; Pelánek, 2017; Sottilare et al., 2016).

It may help to distinguish between a domain model and a student model, another central component of many AIED systems, also known as a learner model. Whereas a domain model captures general domain knowledge, a student model represents an individual student's current learning state. Exactly what state is captured, how it is represented, and how it is kept up to date varies across AIED systems. A student model often captures a student's current level of mastery of the knowledge targeted in the instruction, and may capture other aspects as well (e.g., a student's affective or motivational state, their skill at self-regulation, etc.). To model individual students' knowledge state, the student model is often an "overlay" on the domain model, in the sense that it records the student's status with respect to key elements in the domain model. In other cases, the student and domain model are components within a single integrated model (see Bayesian networks and machine learning paradigms discussed below).

Mainly for purposes of student modeling, it has turned out to be fruitful to view domain models as "knowledge component models," or "KC models" for short (Aleven & Koedinger, 2013; Koedinger et al., 2010). Knowledge components (KCs) are units of knowledge whose existence can be inferred from student performance on a set of related tasks (Koedinger et al., 2012). A KC model breaks up the overall knowledge to be learned into units that reflect students' psychological reality (as evidenced by the fact that they lead to accurate performance predictions). A KC model also maps items (e.g., problem steps, problems) to KCs, indicating which KCs are needed for correct performance on the item. In its simplest form, a KC model is a KC-to-item mapping, or KC x item matrix (a Q-Matrix; Koedinger et al., 2010; Tatsuoka, 1983). This mapping enables a tutoring system to track students' knowledge growth (namely of specific KCs) based on their problem-solving performance. Increasingly, data-driven methods are being used to create and refine KC models (Huang et al., 2021). The KC-modeling perspective is compatible with many of the domain modeling paradigms used in AIED (e.g., rules and constraints can be viewed as KCs learned by students, without considering their specific representation). For more information about student modeling, the interested reader may consult reviews by Desmarais and Baker (2012) and Pelánek (2017).

Although the domain models used in AIED systems are not fundamentally different from those in many other AI systems, they do emphasize certain modeling issues over others. For example, amid the great variety of AIED domain models, there is a premium on having models that capture human ways of reasoning and that can accommodate different ways of reasoning about the same problem. Moreover, it helps if models are interpretable and explainable. An *interpretable* model is one that maps inputs to outputs in a manner that naturally aligns with human ways of understanding or reasoning. An *explainable* model is one for which we have external processes that align the model's state and inferences with human ways of reasoning (Adadi & Berrada, 2018). For example, deep neural networks are not interpretable, but there is work (outside of AIED) on generating natural language explanations that coincide with the classifications of deep neural networks, which bring some transparency to the model (Park et al., 2018). Explainable models can support instructional functions that black box (i.e., non-interpretable) models cannot, such as explaining a reasoning process to learners. Explainable models may also contribute to theory formation more readily. The point that AIED systems both benefit from interpretable and explainable representations of domain knowledge *and* can be a force driving technology design toward greater explainability and interpretability goes back at least to Wenger's (1987) book. It is still true today.

Why Do AIED Systems Have Domain Models?

Before describing different domain modeling paradigms, we consider the many purposes for which domain models are used in AIED systems, to guide later discussion. We look specifically at the adaptive instructional behaviors that they enable or for which they have an auxiliary role, corresponding to the columns in Table 7.1.

Assessing student work

In all the AIED domain modeling paradigms we surveyed, a key function of the domain model is to assess student work, although there is substantial variability in *how*. We use the term "student work" to denote attempted problem solutions, partial solutions, and attempts at problem steps. An assessment may be a determination that the student work is correct, accurate, or of high quality by the standards of the given domain, or it may be based on other, richer classifications of student work's desirable and undesirable qualities, or it may focus on detecting specific qualities or issues. The assessment is often the foundation for other instructional behaviors of the AIED system. Assessment by AIED systems is typically *formative* in nature; its main purpose is to enable and support instructional behavior that helps students improve.

Assessing student knowledge

Domain models often have an auxiliary role in a system's assessment of a student's knowledge growth over time, a central concern in the realm of student modeling. A domain model often helps to analyze student work in terms of KCs. Specifically, given a piece of student work, the domain model helps "diagnose" which KCs the student may have used, mis-used, or mistakenly not used in generating that piece of work. This information is then used to update estimates of the student's knowledge state, using any of the many established student modeling methods or models that have come out of AIED and Educational Data Mining (EDM) research (Desmarais & Baker, 2012; Pavlik et al., 2013; Pelánek, 2017). As discussed, to this end, the student model and domain model often break up the knowledge to be learned into the same set of KCs, as one way in which domain modeling and student modeling tend to be closely interrelated. The domain model and student model are often separate software components, but it is also possible that the domain and student modeling components of an AIED system are integrated together within a single model, as is the case for example in Bayesian networks for stealth assessment (Shute et al., 2016).

Table 7.1 Domain model functions

Paradigms	Assess and model student work	Generate feedback on student work	Demonstrate how to solve problems	Modeling and recognizing student errors	Assess and model student knowledge	Select problems
Rules	Yes (Anderson, Corbett, Koedinger, & Pelletier, 1995)	Yes (Anderson, Corbett, Koedinger, & Pelletier, 1995)	Yes (Anderson, Corbett, Koedinger, & Pelletier, 1995)	Optional (McKendree, 1990)	Auxiliary (Corbett & Anderson, 1995)	Auxiliary (Corbett, McLaughlin, & Scarpinatto, 2000)
Constraints	Yes (Mitrovic & Ohlsson, 1999)	Yes (Mitrovic & Ohlsson, 1999)	No	Contrary to fundamental assumptions	Auxiliary (Mitrovic & Ohlsson, 2016)	Auxiliary (Mitrovic & Ohlsson, 2016)
Generalized Examples (Behavior Graphs)	Yes (Aleven et al., 2016)	Yes (Aleven et al., 2016)	Yes (Aleven et al., 2016)	Optional (Aleven et al., 2016)	Auxiliary (Corbett & Anderson, 1995)	Auxiliary (Corbett, McLaughlin, & Scarpinatto, 2000)
Bayesian Networks	Yes (Conati et al., 2002)	Yes (Conati et al., 2002)	Yes (Conati et al., 2002)	Optional (Stacey et al., 2003)	Yes (Millán & Pérez-de-la Cruz, 2002)	Yes (Mayo & Mitrovic, 2001)
Supervised Learning	Yes (Gobert et al., 2013)	Yes (Li, Gobert, Dickler, & Moussavi, 2018)	Yes (MacLellan & Koedinger, 2020)	Yes (Michalenko, Lan, & Baraniuk, 2017)	Yes (Min et al., 2020)	Open issue
Unsupervised Learning	Yes (Käser & Schwartz, 2020)	Open issue	Open issue	Yes (Shi et al., 2021)	Open issue	Auxiliary
Reinforcement Learning	Yes (Rafferty et al., 2015)	Yes (Rafferty, Jansen, & Griffiths, 2016)	Yes (Barnes & Stamper, 2008)	Auxiliary (Barnes & Stamper, 2008)	Yes (Raffery et al., 2015)	Yes (Beck, Woolf, & Beal, 2000)

Providing formative feedback to students

A key function of many AIED systems is to give students feedback on their work, for example as part of a coaching strategy (Chapter 9 by Aleven et al.). The purpose of this feedback is typically to help students learn; in this sense, the feedback is "formative" (Shute, 2008). The notion of feedback is often divided into three main categories: correctness feedback, knowledge of results, and elaborated feedback (Kluger & DeNisi, 1996; van der Kleij et al., 2015). Correctness feedback indicates whether the work is correct or not, or it signals the degree of correctness. Knowledge of results means providing a correct answer or solution. Elaborated feedback provides further information regarding correct/incorrect aspects. The latter category of feedback may state desirable and undesirable properties of the student's work, or how it might be improved. Generating feedback is facilitated by having an interpretable or explainable domain model, as discussed below.

Recognizing student errors

AIED systems sometimes provide *error-specific feedback*, a form of elaborated feedback that comments on specific errors or misconceptions reflected in the student's work. The feedback might say, for example, how or why student work is wrong, or how an error might be fixed. One way to generate error-specific feedback in an AIED system is to model common erroneous knowledge into the system's domain model (e.g., bugs, mal-rules, and misconceptions). The empirical evidence regarding the pedagogical value of error-specific feedback, however, is not elaborate and is mixed (Lodder et al., 2021; McKendree, 1990; Sleeman et al., 1989; VanLehn, 1990; VanLehn et al., 2021), which is perhaps a key reason that such feedback tends to be treated as optional in AIED systems (see Table 7.1). Then again, error-specific feedback is often viewed as an attractive feature of AIED systems. Thus, there is more to be learned in the field of AIED about the topic of error-specific feedback.

Demonstrate how to solve problems to provide next-step hints

Some AIED systems can use their domain models to demonstrate problem solutions (i.e., the domain model can solve problems), which enables them to generate next-step hints for students. These hints suggest what the student might do next and may include information such as how in general one might determine the next step and why that is a correct thing to do (e.g., in terms of domain-specific principles). Hints may be offered either at the student's request or proactively by the system. Their purpose is to help students better acquire the knowledge to be learned and to help them avoid floundering (e.g., help them avoid a fruitless search for a solution or step when they lack the knowledge). Next-step hints are different from feedback in that they do not provide an assessment of student work.

Selecting individualized learning content

The domain model often plays an auxiliary role in another key function of AIED systems, namely selecting or sequencing learning content (e.g., problems, tasks, textbook pages) for students on an individualized basis. This choice is often based on variables in the student model, in particular students' mastery of knowledge components, sometimes in combination with other variables (e.g., personal interest, affect). The content selection process or algorithm is outside of the domain model per se; it is viewed as part of a pedagogical model. The domain model helps in the first place by assessing student knowledge, as described above. As well, some domain models represent prerequisite relations between knowledge components, so that a selection algorithm can sequence learning materials in accordance with these relations.

Designing new learning content

A final important function of domain models is to guide the design and redesign of content for AIED systems. A domain model can help with many aspects of content including inventing new tasks (Huang et al., 2021; Koedinger & McLaughlin, 2010; Koedinger et al., 2013), new hint messages (Liu & Koedinger, 2017) or new contrasting cases (Roll et al., 2010; Schwartz et al., 2011). An initial KC model (based, ideally, on empirical cognitive task analysis) can guide designers in creating initial versions of these content elements. Iterations may follow once cohorts of students have used the system and log data are available to feed data-driven approaches to KC model refinement. As a simple example, the pool of problems that an AIED system can assign to students must provide sufficient coverage of each KC, so that students can get sufficient practice with each. Similarly, the system's hints will likely be designed to closely track the KCs (e.g., with hint templates attached to specific KCs). When the KC model changes (e.g., when a process of data-driven model refinement discovers new KCs), these aspects need to be revised and a refined model can provide much guidance (see Huang et al., 2021).

DOMAIN MODELING PARADIGMS

We review four major paradigms for domain modeling in AIED systems: rules, constraints, Bayesian networks, and machine learning. We also briefly describe several other paradigms: labeled example solutions (including behavior graphs), knowledge spaces, and domain modeling in textbooks.

Rules

Production rules remain a popular formalism for representing domain knowledge in AIED systems. The use of rule-based models in AIED systems is grounded both in AI work on production rule systems (Brownston et al., 1985; Davis & King, 1984) and in cognitive science work that uses rules to represent aspects of human cognition and human problem solving (Anderson, 1993; Newell & Simon, 1972). In this knowledge representation paradigm, domain knowledge is expressed as a set of IF-THEN rules. Each rule ties one or more (mental or observable) problem-solving actions (the THEN-part) to the conditions under which they are appropriate (the IF-part). Rule-based models, which may comprise hundreds of rules, are executable and capable of solving problems in the given task domain. They can be viewed as simulations of expert (and student) problem solving in the given task domain. Rule-based models used in AIED systems typically represent the knowledge that the system is designed to help students learn. For simple examples, see Koedinger and Corbett (2006) and Aleven (2010). For a more elaborate example, see Aleven (2010).

Rule-based domain models have been used extensively in model-tracing tutors, a widely used type of AIED system grounded in cognitive science and cognitive modeling (Anderson et al., 1995). Model-tracing tutors guide students as they solve complex problems, that is, problems that have multiple possible solution paths, each with multiple steps. Many model-tracing tutors have been described in the AIED literature, including Cognitive Tutors for middle-school and high-school mathematics (Koedinger & Corbett, 2006), the Genetics Tutor (Corbett et al., 2010), Cognitive Tutors for Lisp, Pascal, and Prolog programming (Anderson et al., 1989; Anderson et al., 1993), Lynnette (middle-school equation solving;

Long & Aleven, 2014), MATHia (middle- and high-school mathematics; Ritter et al., 2007), Andes (physics; VanLehn et al., 2005), SlideTutor (skin pathology; Crowley & Medvedeva, 2006), and MATHESIS (high-school algebra; Sklavakis & Refanidis, 2013). Model-tracing tutors for mathematics learning are being used widely in American mathematics learning (Ritter et al., 2007). There is substantial evidence that model-tracing tutors can help students learn very effectively (for an overview, see Koedinger & Aleven, 2007), including a large-scale study that found a doubling of the amount of learning within a school year due to the Cognitive Tutor Algebra curriculum (Pane et al., 2014; but also see Pane et al., 2010).

Model tracing tutors use their rule-based models for many of the functions described above: assessing student work, providing hints and feedback, interpreting student problem solving in terms of knowledge components, and guiding content design. To support these functions, model-tracing tutors use their rule-based model to maintain a live, up-to-date, step-by-step reconstruction of a student's problem-solving process, in sync with the student's solution (as the student is working with the tutoring system). For this approach to work, the model must capture *all* reasonable ways of solving problems that students might use, one way in which rule-based models used in AIED systems differ from those used in many other AI applications. A model-tracing tutor can also use its domain model to generate next-step hints at any point in a student's problem-solving process. To do so, the tutor finds an applicable rule (i.e., one that could generate the next step from the current problem state) and generates an explanation of that step (and why that is a correct or good step to take) using the rule's hint template. For this approach to yield understandable next-step advice, the rules must capture human approaches to problem solving. In this sense, the rules must be explainable. From the perspective of Cognitive Tutors, rules are a key analytical tool for understanding student reasoning and learning in a given task domain. They can be used to summarize results of cognitive task analysis activities, which can be an important step in designing an intelligent tutoring system (Baker et al., 2007; Lovett, 1998; Means & Gott, 1988; Tofel-Grehl & Feldon, 2013). The model-tracing process as described feeds into the tutor's long-term student modeling process. Following any student step, it identifies which rule(s) a student applied or should have applied but did not. This information enables the tutor to track the probability that the given student masters each of the key rules in the model, for example using a model such as Bayesian Knowledge Tracing (Corbett & Anderson, 1995).

We illustrate the main idea behind model tracing with a simple example provided by Ken Koedinger. As mentioned, a key challenge in model tracing is that any given problem may be solved in multiple ways, captured in the model. Of these many solution paths, the model must follow the one that the given student is using for the given problem. It does so in a step-by-step manner. For example, even basic equations (such as the one in Figure 7.1) can be solved in different ways. Assume a model for basic equation solving with the three rules shown at the top of Figure 7.1. Two of the rules represent correct student strategies, and one represents an erroneous strategy. Note all three rules have the same IF-part, so they apply to the same set of problem states. The letters a, b, c, and d represent variables that can match specific numbers in the problem state. Whenever the student attempts the next problem step, the model tracer searches the space of possible next steps generated by its rule model to see if the student step is among the model-generated steps. In our example, in the given problem state (i.e., $3(2x - 5) = 9$), our three rules apply. The model tracer (searching through the set of applicable rules) will find three possible next steps, namely two correct steps and an error, shown at

Strategy 1:	IF the goal is to solve a(b*x* + c) = d THEN rewrite this as ab*x* + ac = d
Strategy 2:	IF the goal is to solve a(b*x* + c) = d THEN rewrite this as b*x* + c = d/a
Misconception:	IF the goal is to solve a(b*x* + c) = d THEN rewrite this as ab*x* + c = d

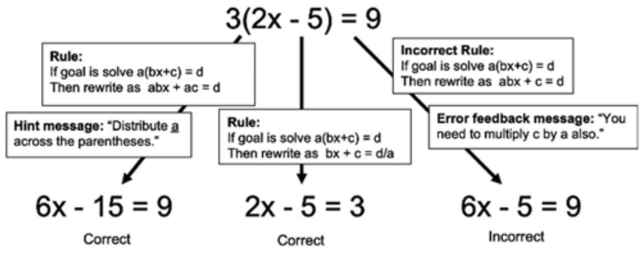

3(2x - 5) = 9
Rule: If goal is solve a(bx+c) = d Then rewrite as abx + ac = d
Hint message: "Distribute a across the parentheses."
Rule: If goal is solve a(bx+c) = d Then rewrite as bx + c = d/a
Incorrect Rule: If goal is solve a(bx+c) = d Then rewrite as abx + c = d
Error feedback message: "You need to multiply c by a also."
6x - 15 = 9
Correct
2x - 5 = 3
Correct
6x - 5 = 9
Incorrect

Figure 7.1 Model tracing in action

the bottom of Figure 7.1. If the student's input is one of the correct transformations, the tutor accepts it as correct and applies the matching rule to move the state of the rule-based model forward. By contrast, if the student's input is the incorrect transformation (i.e., $6x - 5 = 9$), the tutoring system presents an error message based on the erroneous rule, generated using a hint template attached to the rule. If the student's input is anything else, the tutor flags it as incorrect, without specific error feedback. This way, the model stays in sync with the student. Finally, if the student requests a hint in the given situation, the tutor will recommend a step generated by one of the rules (the one with highest priority), again using a template attached to the rule. Incidentally, rules that represent incorrect problem-solving behavior are not strictly required in model-tracing tutors. They enable the tutoring system to provide elaborated error-specific feedback.

This example is a simplification in that the model has very few rules that apply to only a narrow range of problem states. The rule-based models in model-tracing tutors, by contrast, can have hundreds of rules that apply to a wide range of problem states. A second way in which this example is a simplification is that each problem-solving step is modeled by a single rule, whereas, in the more general case, a problem-solving step may result from a *sequence* of rule applications—any number of them, in fact. In the general case, therefore, the model tracer has more searching to do to find the possible model-generated next steps against which to compare the student's next step.

Over the years, rule-based models in AIED have been used for purposes other than modeling domain knowledge to be learned. For example, rule-based models have been used to capture—and provide tutoring regarding—strategy aspects of (algebra) problem solving (Ritter, 1997), aspects of self-regulated learning, such as help seeking (Aleven et al., 2006a) and error correction (Mathan & Koedinger, 2005), as well as collaboration skills (Walker et al., 2014). Various projects have also used rules to model pedagogical knowledge (Aleven et al., 2017; Heffernan et al., 2008; Roll et al., 2010), illustrating the versatility of rules as a representational paradigm for AIED systems.

Challenges

A key challenge is that it is hard to create rule-based models of problem solving. This process requires cognitive task analysis, cognitive modeling, and AI programming. To address this challenge, a considerable amount of AIED work has focused on developing efficient authoring tools (see Chapter 12 by Blessing et al.), use of interactive machine learning to develop rule-based models (see below), and non-programmer AIED paradigms that can achieve some of the same tutoring behaviors with easier-to-create knowledge representations (e.g., example-tracing tutors [Aleven et al., 2016]; see below). This work is still moving forward. A second critique that is sometimes leveraged against tutoring systems with rule-based models is that they might be limited to STEM domains or domains with clear correctness criteria. Although many AIED projects with rule-based domain models have indeed focused on STEM domains, other domains have been explored as well (e.g., medical diagnosis; Crowley & Medvedeva, 2006). As well, clear correctness criteria can sometimes be identified in domains not initially thought to be amenable to rule-based modeling (Means & Gott, 1988; Tofel-Grehl & Feldon, 2013). It may be challenging, however, to create model-tracing tutors in ill-defined domains (Lynch et al., 2009) or domains with natural language interactions (see Chapter 11 by Rus et al.). A third critique has been that the pedagogical approach of model-tracing tutors (tutored problem solving) tends to be limited. While this critique may not fully value the important role of deliberate practice and learning to solve recurrent problems in many domains (Koedinger & Aleven, 2021), it is important to note that rule-based tutors are often combined with other instructional approaches, such as example-based learning, collaborative learning or standard classroom instruction (e.g., Koedinger & Corbett, 2006; Olsen et al., 2019; Salden et al., 2010).

Constraints

Another popular formalism for representing domain knowledge in AIED systems is constraints (Mitrovic, 2010; Mitrovic & Ohlsson, 2006; Ohlsson & Mitrovic, 2007). In Constraint-Based Modeling (CBM), the domain model consists of a set of constraints on ideal solutions. In comparison to rule-based cognitive models, which capture procedural knowledge, constraint-based domain models capture the declarative knowledge of a specific instructional domain. Constraints capture features of correct solutions; they specify what ought to be so.

Numerous constraint-based tutors have been developed, some of which are SQL-Tutor, an ITS for the Structured Query Language (SQL) (Mitrovic, 1998; Mitrovic & Ohlsson, 1999), and EER-Tutor, an ITS teaching conceptual database design using the Enhanced Entity-Relationship model (EER)(Mitrovic, 2012). Constraints have not only been used to represent domain principles, but also to model collaboration (Baghaei et al., 2007) and metacognitive strategies such as self-explanation (Weerasinghe & Mitrovic, 2006), and have even been used for rehabilitation of prospective memory of stroke patients (Mitrovic et al., 2016).

The theoretical foundation for CBM comes from Ohlsson's Theory of Learning from Performance Errors (Ohlsson, 1996). This theory says that both declarative and procedural knowledge is necessary for good performance, but the theory focuses on declarative knowledge, represented as a set of constraints on solutions. People make mistakes when they do not have adequate procedural knowledge (either missing or incorrect). Constraints are used to identify mistakes and to repair incorrect production rules. This theory explains how it is possible for a person to know that he/she made a mistake even though they do not have correct procedural knowledge. The role of conceptual knowledge is to identify mistakes.

A constraint consists of two conditions. The relevance condition consists of one or more tests applied to students' solutions to see whether a constraint is relevant or not. If a constraint is relevant for a particular solution, its satisfaction condition specifies another list of tests which the solution must meet to be correct. An example of a constraint is: if you are driving a car in New Zealand, you should be on the left-hand side of the road. The relevance condition specifies that this constraint is applicable to situations when a person is driving a car in New Zealand; the satisfaction condition imposes a test on the side of the road the person is driving on. Correct solutions violate no constraints. Constraint violation signals errors in the solution.

We provide some constraints from SQL-Tutor, which contains 700+ constraints related to the use of the SQL Select statement. We present the constraints in the English form; the interested reader is referred to Mitrovic (1998), Mitrovic (2003) and Mitrovic and Ohlsson (1999) to see how the constraints were implemented in Lisp.

1) Every solution must contain the SELECT clause.
2) Every solution must contain the FROM clause.
3) If the solution contains the HAVING clause, the GROUP BY clause also needs to be specified.
4) If the solution contains the JOIN keyword, the FROM clause must specify the names of tables to be joined.
5) If the FROM clause of the student's solution contains a join condition, and the ideal solution requires the join condition between the same two tables, the join condition the student specified needs to use the correct join attributes.
6) If the ideal solution contains a search condition using the Between predicate and two constants, the student's solution should also contain a matching condition or alternative two conditions, using the same attribute and corresponding constants.

Some constraints are syntactic (such as constraints 1–4), meaning they check the syntax of the submitted Select statement. Constraints 1 and 2 do not contain the "if" part; they are relevant for all solutions, as the SELECT and FROM clauses are mandatory in SQL. On the other hand, constraints 3 and 4 have relevance conditions that restrict the set of solutions for which these constraints are relevant. If the relevance condition is met, the satisfaction condition is evaluated against the solution.

SQL-Tutor also contains semantic constraints (see example constraints 5 and 6); these constraints check whether the submitted solution is a correct solution for the particular problem. The semantics of the solution is captured by the ideal solution, which is defined by the teacher. Although many problems in SQL have multiple correct solutions, only one ideal solution is stored in SQL-Tutor per problem; the other correct solutions are recognized by the system automatically as there are constraints that check for alternative ways of solving the same problem (Mitrovic, 2003). This makes adding new problems very simple: the author needs to provide the text of the problem and one correct solution only.

Semantic constraints check whether the student's solution is correct by matching it to the constraints and the ideal solution. For example, constraint 5 checks that the student has used the correct attributes to join two tables which are also used in the ideal solution. Constraint 6, on the other hand, is relevant for those problems that require a search condition checking that the value of an attribute is between two specified values; for example, a range search like "Year between 2000 and 2022." For the student's solution to be correct, it should also contain a matching range search; alternative correct solutions need to contain two comparison conditions using the correct constants (e.g., "Year > 1999 and Year < 2023").

The space of correct knowledge can be big, but the space of incorrect knowledge is huge. CBM does not model incorrect knowledge; on the contrary, constraints only capture features of correct solutions. The abstraction used is the sets of equivalent states of the problem space. Each equivalence set corresponds to one constraint, which specifies one aspect of a domain principle; basically, an equivalence set of problem states is the set of solutions which all use the same domain principle. The whole set of states is represented via a single constraint. All solutions that require that constraint must satisfy it to be correct. Otherwise, there is a mistake in the student's solution. This makes CBM applicable to ill-defined tasks, such as design (Mitrovic & Weerasinghe, 2009). The domain model in such cases captures what is known about good solutions; the aspects which are not crucial are not checked. We refer to CBM as the "innocent until proven guilty" approach; if the student's solution does not violate any constraints, it is deemed correct.

In constraint-based tutors, the student model is represented as an overlay on the constraint set. The short-term student model is the result of diagnosing the solution the student submitted and consists of the set of satisfied and potentially violated constraints. The long-term model of the student's knowledge is represented in terms of the student's knowledge of individual constraints. There have also been extensions of the long-term model using Bayesian networks (Mayo & Mitrovic, 2001).

Constraints are evaluative in nature, as opposed to production rules which are generative (i.e., each rule generates an action to be performed to solve the problem). Constraints are modular and can be applied in parallel to diagnose the student's solution. This diagnosis consists of matching the student's solution to the constraint set and the ideal solution. The result is the set of matched constraints and, if the solution contains mistakes, a set of violated constraints. Each violated constraint represents a tiny part of the domain and is used to generate feedback for the student. In constraint-based tutors, each constraint typically has one or more feedback messages. These messages can be used to provide negative feedback (i.e., feedback on errors), with a gradually increasing level of detail. Messages attached to constraints can also be used to provide positive feedback, which are given to the student when the student masters a new piece of knowledge (e.g., using the constraint correctly for the first time), when the student overcomes impasses (e.g., satisfying a constraint after making a series of mistakes), or when the student solves a challenging problem (Mitrovic et al., 2013).

All constraint-based tutors match the student's solution to the constraints and the ideal solution. Depending on the instructional domain, the ideal solution may need to be stored (in the case of design tasks, where there is no problem solver), or it can be generated on the fly (for procedural tasks). When developing constraints, the level of granularity is crucial; constraints need to be very specific, so that feedback messages can be useful. If constraints are written on a very abstract level, the feedback messages would not be useful to students. Our general advice for writing constraints is to think about what a human teacher would say to the student if the constraint is violated. In addition to writing constraints manually, there is also ASPIRE, an authoring system (Mitrovic et al., 2009; Suraweera et al., 2010). ASPIRE requires the teacher to specify a simple domain ontology, the structure of solutions in terms of ontology concepts, and to provide examples of solved problems. Based on that information, ASPIRE induces a set of constraints for evaluating students' answers (see Chapter 12 by Blessing et al.).

Challenges

Like rule-based AIED systems, the key challenge for constraint-based tutors is the development of the constraint set. This process requires domain knowledge, pedagogical expertise (for specifying the hint messages attached to constraints) and AI programming. The ASPIRE authoring system (Mitrovic et al., 2009) automates a lot of the functionality for generating a constraint set, but pedagogical expertise is still necessary.

Bayesian Networks

Bayesian Networks (BNs) are a powerful AI formalism for reasoning with uncertainty, combining principles from graph theory and probability theory (Pearl, 1988; Russell & Norvig, 2020). BNs have been used in a broad range of task domains (e.g., medical diagnosis), and became actively used in the AIED field for domain modeling and student modeling in the early 1990s. A BN is a probabilistic model, often depicted as a directed graph, where nodes represent random variables and directed edges between the nodes represent dependencies between the variables. Central to BNs is the notion of conditional independence, which allows the joint probability of variables to be computed by multiplying the conditional probabilities of each variable given its parents. An attractive property of BNs is that they allow for two kinds of reasoning with uncertainty within a single integrated framework: diagnosis (i.e., identifying likely causes, given effect observations), and prediction (i.e., predicting the likely effect, given the current beliefs of the causes). In AIED systems, diagnosis typically involves inferring a student's knowledge of KCs (or other latent states, such as goals or plans) targeted in the instruction, given the student's performance. Diagnosis is instrumental to assessment and student modeling. Prediction involves predicting a student's performance based on the current estimates of the student's knowledge levels, which could be used to realize functions such as adaptive content selection or hint selection. In both forms of reasoning, uncertainty arises due to noise in student behaviors (e.g., guessing without knowing the KC) and noise in domain model specifications (e.g., a KC not being identified).

BNs used in AIED support a range of functionality, including knowledge assessment or student modeling (Millán & Pérez-de-la Cruz, 2002), modeling students' misconceptions (Stacey et al., 2003), solution plan recognition (Conati et al., 2002), error or hint message selection (Conati et al., 2002; Mayo & Mitrovic, 2001) and problem selection (Ganeshan et al., 2000; Huang, 2018; Mayo & Mitrovic, 2001). The BN paradigm is a good choice for domain modeling and student modeling for several reasons. First, BNs handle reasoning with uncertainty based on sound mathematical theories. Second, BNs afford high expressiveness. They can be used to create models that integrate domain knowledge with other kinds of student variables (e.g., metacognitive skills and affective states). Third, BNs naturally integrate domain modeling, student modeling, and aspects of pedagogical modeling in a single framework. Finally, BNs provide a principled way to utilize prior domain knowledge and integrate new data with prior knowledge. A more thorough introduction can be found elsewhere (Millán et al., 2010).

There have been many successful cases of utilizing BNs in domain modeling for AIED systems. One prominent example is the Andes tutoring system for physics (Conati et al., 2002). In Andes, a BN that integrates domain modeling and student modeling enables step-by-step problem-solving support tailored to each student's knowledge and strategy choices. As shown in Figure 7.2, Andes' BN consists of a domain-general component that models students' long-term knowledge shared across problems, and a task-specific component that models possible correct solution paths specific to each problem. In the domain-general component, *Rule* nodes

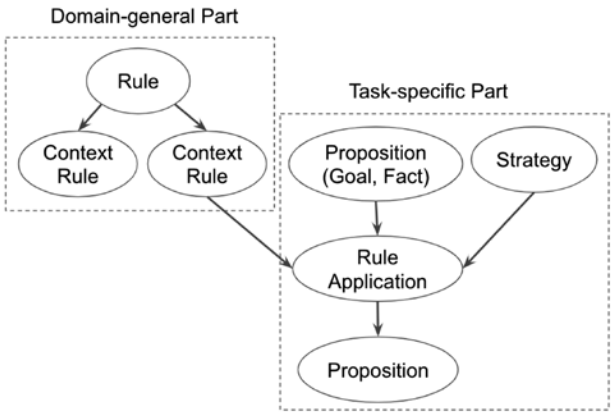


Figure 7.2 The basic structure of Andes' BN-based domain and student model

represent knowledge in a general form (e.g., being able to apply F=ma in all possible contexts) while *Context Rule* nodes represent knowledge in context-specific forms (e.g., being able to apply F=ma in a specific context). In the task-specific component of the BN, a *Rule Application* node represents an application of a specific rule (KC) in a problem (e.g., the application of F=ma in the current problem). A *Proposition* node represents a goal (e.g., try F=ma to solve the problem) or a fact (e.g., block A is selected as the body). *Strategy* nodes, finally, can be used to model different, mutually exclusive, correct solutions. *Rule Application* nodes connect *Context Rule* nodes, *Proposition* nodes, and *Strategy* nodes to newly derived *Proposition* nodes. All nodes can have true or false values. Depending on the node type, the probability of a node taking on a true value represents a student's knowledge level or the likelihood that the given student will either infer a goal/fact or choose a strategy. Uncertainty is handled by assuming a small probability of slipping or guessing in the conditional probabilities for *Rule Application* or *Proposition* nodes. Once the structure and parameters of Andes' BN have been specified (expert engineered), the BN is used to estimate how likely it is that a student can derive a goal/fact, will choose a specific strategy, or will apply a specific rule (KC). The BN updates these estimations after each observed student action. These estimations drive tutorial interventions such as selecting a hint topic, providing mini lessons for weakly mastered KCs, and selecting problems with desirable difficulties. For example, when a student requests a hint, Andes figures out what goal the student is likely trying to achieve (i.e., plan recognition) by comparing the probabilities of related goals for the most recent student action. It then looks for a related rule application with a low probability and finally makes the relevant knowledge the focus of the hint.

Classroom evaluations of Andes demonstrated that students who used Andes for their homework significantly improved their learning compared with students who used pencil and paper for their homework, reaching effect sizes of 1.2 (conceptual components) and 0.7 (algebraic components) on experimenter-designed tests and 0.3 on standardized tests (VanLehn et al., 2005). VanLehn et al. (2005) pointed out that Andes' key feature appears to be the grain-size of interaction (i.e., on a step level rather than on a problem level). Other examples of the use of BNs in AIED systems include the CAPIT tutor for English capitalization and punctuation (Mayo & Mitrovic, 2001) and Cognitive Tutors for algebra (Ritter et al., 2007), where classroom evaluations demonstrated that systems using BN-based domain modeling (and student modeling) yielded greater student learning outcomes or efficiency compared with systems that used alternative modeling or traditional classroom instruction.

We classify existing cases of utilizing BNs in AIED systems into four types, based on several past reviews (Desmarais & Baker, 2012; Millán et al., 2010; Pelánek, 2017): *Prerequisite, Granularity, Solution, and Independence*. In Prerequisite BNs, the edges between nodes denote prerequisite relations between KCs. Such networks can help better sequence learning materials and increase the efficiency and accuracy of knowledge assessment (Carmona et al., 2005; Käser et al., 2014; Reye, 1996). In Granularity BNs, nodes and edges are organized in a hierarchy to decompose domain knowledge into different levels of detail (e.g., a topic and a subtopic). They allow for knowledge assessment at different levels. Granularity BNs have been investigated in several prior projects (Collins et al., 1996; Millán & Pérez-de-la Cruz, 2002; Mislevy & Gitomer, 1995). Another type is Solution BNs, which represent problem solution paths in conjunction with knowledge levels. For example, Andes' task-specific part of the BN corresponds to a solution graph composed of goals, facts, rule applications, context rules, and strategies, allowing for knowledge assessment and plan recognition for generating hints and instructions (Conati et al., 2002). In Independence BNs, independence among KCs is assumed. An example is Bayesian Knowledge Tracing (BKT; Corbett & Anderson, 1995) where knowledge estimation of a KC is independent of that of other KCs. BKT models knowledge dynamically with a dynamic Bayesian network (DBN) with key structure repeated at each time slice and with additional edges connecting the same types of knowledge nodes across time slices. Another kind of Independence BN uses a flat structure in which there is only one layer of KCs and there are no edges among KCs (Huang et al., 2017).

To reduce the number of parameters for ease of BN construction and inference, logic gates (Noisy-AND or Leaky-OR) can be used (Conati et al., 2002; Huang et al., 2017). There are other variations, such as integration with decision theory (Mayo & Mitrovic, 2001; Murray et al., 2004), complex dynamic BNs in narrative-centered learning environments (Rowe & Lester, 2010), logistic regression models for efficient modeling of subskills (González-Brenes et al., 2014; Xu & Mostow, 2012), and modeling integrative KCs that integrate or must be integrated with other KCs to produce behaviors (Huang et al., 2017).

There are three main approaches for creating or refining a BN-based domain (and student) model: expert-engineered, automated, or mixed approaches. (Mayo and Mitrovic (2001) use the terms expert-centric, data-centric, and efficiency-centric approaches, respectively.) Most of the work in AIED falls into the category of expert-engineered approaches where an expert specifies directly or indirectly the structure and parameters of the BN (Conati et al., 2002; Mislevy & Gitomer, 1995). Some research took an automated approach where the BN structure and parameters are learned primarily from data (Chen et al., 2016; Mayo & Mitrovic, 2001). Other research applied a mixed approach, where several BNs are specified first based on domain knowledge and then compared in terms of predictive accuracy on collected data (Pardos, Heffernan, Anderson, & Heffernan, 2006), or a BN is partially learned from data and then refined by experts (Vomlel, 2004). To evaluate a BN for domain modeling, data-driven evaluations on simulated datasets (Conati et al., 2002; Mayo & Mitrovic, 2001; Millán & Pérez-de-la Cruz, 2002) or real-world datasets (Huang, 2018; Pardos et al., 2006), as well as classroom studies (Conati et al., 2002; Huang, 2018; Mayo & Mitrovic, 2001), have been conducted.

Challenges

There are several concerns or challenges regarding the use of the BN paradigm for domain modeling in AIED systems. In cases where BNs are constructed by expert engineering, the process can be time-consuming and error-prone. Although many data-driven approaches have been devised to address this issue, fully automated methods for learning BNs (structure or parameters) in many cases are still computationally expensive and require a substantial amount of data to reach acceptable accuracy (Millán et al., 2010). Several issues that are relevant to many machine learning models also apply here. One is the model degeneracy issue where parameters learned from data conflict with the model's conceptual meaning, such as a student being more likely to get a correct answer if they do not know a skill than if they do (Baker et al., 2008; Huang, 2018; Huang et al., 2015). Another is the identifiability issue where the same data can be fit equally well by different parameters, resulting in different system behaviors (Beck & Chang, 2007; Huang et al., 2015), or different interpretations of effects of system features (Huang et al., 2015). To address this, constraints or prior distribution of parameters could be imposed when fitting parameters (Beck & Chang, 2007; Huang, 2018). More elaboration of these issues can be found in *Challenges* in the next machine learning section.

Machine Learning

Machine learning (ML) techniques are widely used in AIED systems, and they play an important role in domain modeling (Koedinger et al., 2013). Machine learning and domain representations intersect in two primary ways. First, domain knowledge is often encoded in the input representations used by machine learning models in AIED systems. These representations of domain knowledge are used to enhance the models' predictive effectiveness across a range of AIED tasks, such as assessing student knowledge (González-Brenes et al., 2014; Min et al., 2020), recognizing student affect (Jiang et al., 2018) or making pedagogical decisions (Rowe & Lester, 2015; Shen, Mostafavi et al., 2018), among others. In many cases, these applications are distinct from the task of modeling knowledge in a given domain itself. When data are provided to a machine learning algorithm, they are typically encoded using a factored representation known as a feature vector (Russell & Norvig, 2020). The attributes in this feature vector representation may indirectly encode information about knowledge in the domain, such as the current problem-solving context or characteristics of expert problem-solving behavior (Geden et al., 2021; Gobert et al., Baker, 2013; Rowe & Lester, 2015). The input feature representations utilized by machine learning models can be either manually engineered or learned automatically as typified by applications of deep neural networks (Jiang et al., 2018; Min et al., 2016).

The second way in which ML and domain representations intersect is in using ML to create or refine a model of domain knowledge itself. This latter approach encompasses a range of different modeling tasks within AIED systems. For example, ML techniques have been used to model effective inquiry strategies (Gobert et al., 2013; Käser & Schwartz, 2020), discover or refine skill models (Boros et al., 2013; Cen et al., 2006; Desmarais & Naceur, 2013; Lindsey et al., 2014; Huang et al., 2021), detect student misconceptions (Michalenko et al., 2017; Shi et al., 2021), and generate automated next-step hints based upon prior students' learning behaviors (Barnes & Stamper, 2008).

Several families of ML algorithms have been examined for domain modeling in AIED systems, including supervised learning, unsupervised learning, and reinforcement learning techniques. We briefly discuss each of these families and provide examples showing how they have been used for domain modeling in AIED. An issue that merits acknowledgment is the relationship between ML approaches for *domain modeling* and *student modeling*. These tasks are often closely related. In some cases, they may coexist within a single model, as we already saw with Bayesian Networks, described in the previous section. In the current section, we focus on work that applies ML to capture data-driven models of target knowledge, skills, and strategies in a particular task domain and/or learning environment. As discussed previously, we distinguish this from work investigating models that capture students' current learning states, which we regard as student modeling.

Supervised learning

Supervised learning is a family of ML techniques that involve training a model on labeled data in order to classify or predict the outcome associated with a new, as yet unseen input (Bishop, 2006; Russell & Norvig, 2020). Supervised learning encompasses a range of algorithmic techniques, including linear models, decision trees, kernel machines, probabilistic graphical models, deep neural networks, and ensemble techniques. A related paradigm is semi-supervised learning, which augments the supervised learning process by combining a small amount of labeled data with a large amount of unlabeled data during training in order to improve model predictions (Van Engelen & Hoos, 2020). Semi-supervised learning has been used to identify learning outcomes and prerequisites from educational texts (Labutov et al., 2017) and predict student performance on assessments (Livieris et al., 2019).

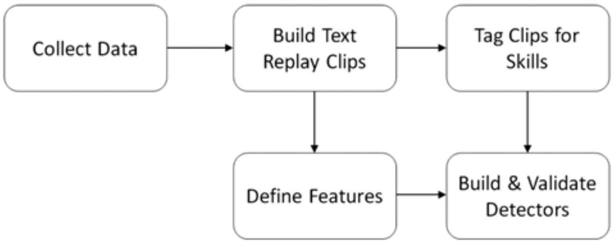
An important domain modeling task is automated model discovery (Cen et al., 2006; Chaplot et al., 2018; Koedinger et al., 2012). Automated model discovery is a form of cognitive model optimization that applies data-driven techniques to refine computational representations of knowledge and skills targeted for learning in a particular domain. For example, Learning Factors Analysis (LFA) is a semi-automated method for refining a cognitive domain model that combines multiple logistic regression and combinatorial search (Cen et al., 2006). The fit of a multiple logistic regression model to students' problem-solving data serves as a heuristic to guide a search for an improved domain representation. The search involves iteratively decomposing problem-solving skills (KCs) based on expert-defined difficulty factors to obtain a refined domain model that better fits the student data. In LFA, supervised learning supports a procedure for discovering (or refining) what the KCs in a domain model *are*. Supervised learning is not used to directly infer relations between KCs and/or student actions, but rather to guide a search process that reveals them. LFA and related techniques have been investigated with a range of algorithmic variations and AIED systems (Koedinger et al., 2012; Huang et al., 2021).

Supervised learning has also been used to model domain-specific strategies by training models with annotated datasets consisting of student learning interactions. Gobert and colleagues used supervised learning to devise models of students' inquiry processes in the Inq-ITS intelligent tutoring system (Gobert et al., 2013). Specifically, they utilized a combination of text replay tagging and decision tree classifiers to create ML-based detectors for automatically recognizing when students demonstrate the skill of designing controlled experiments within science microworlds. Similar techniques have also been used to model other facets of student interaction with AIED systems, such as detecting student affect (Jiang et al., 2018) and gaming the system (Baker et al., 2010; Paquette & Baker, 2019), which highlight the close connection between ML approaches to domain modeling and student modeling.

Work by Gobert et al. (2013) offers an example of how supervised learning can be used to create a domain model to assess student inquiry processes. (Their Inq-ITS system, on which this example is based, is mentioned briefly in the section "Exploratory Learning" of Chapter 9 (Aleven et al.).) Figure 7.3 provides a high-level illustration of the major steps involved. The first step is collecting log data from student interactions with an AIED system. Gobert and colleagues use data from 148 eighth-grade students collected during a classroom implementation of the Inq-ITS web-based tutor. In Inq-ITS's phase change microworld, students are given the task of determining which variables (e.g., container size, heat level, substance amount) affect different properties of a substance's phase change. Students proceed through a series of inquiry phases—exploring the simulation, using built-in tools to generate hypotheses, conducting simulated experiments, analyzing data—to complete the task. Students' interactions with Inq-ITS are time-stamped and logged to produce a low-level, sequential record of students' inquiry behaviors in the science microworld. Next, students' logs are segmented into meaningful sets of student actions, called clips, to be annotated by human coders. Each clip is tagged with one of ten possible inquiry skill labels, capturing productive inquiry behaviors such as "Designed Controlled Experiments," "Tested Stated Hypothesis," and "Used Data Table to Plan," as well as undesirable inquiry behaviors such as "Never Changed Variables" and "No Activity" (Sao Pedro et al., 2013). Two coders each assign a single label to each clip for a subset of clips. The subset of clips is used to establish interrater agreement, and, after establishing close agreement, the coders split up the remaining clips to be coded separately.

Next, the student log data is processed to distill features that will serve as input for machine learning. Gobert et al. (2013) distilled 12 predictor features from the student log data for each tagged clip: *all actions count*, *complete trials count*, *total trials count*, and so forth. For each tagged clip, the predictor features are combined into a vector with the "ground truth labels" from human coders appended. The set of feature vectors and their associated labels serve as the dataset for creating the machine learning model to detect student design of controlled experiments.

The dataset is then split into separate training, validation, and test sets. A widely used approach for model evaluation in supervised learning is called *cross-validation*, which involves repeatedly splitting a dataset into separate subsets for training and validation and alternately using each data point as either training data or validation data (Russell & Norvig, 2020). In cross-validation, training sets are used to tune the machine learning model's parameters, and



Source: Adapted from Gobert et al. (2013)

Figure 7.3 *Steps for creating a domain model to detect the skill of designing controlled experiments using supervised learning*

validation sets are used to test the model's predictive performance on data not used in training. In the work of Gobert and colleagues (Gobert et al., 2013), the training data was provided as input to a J48 decision tree algorithm (Quinlan, 1993). In this algorithm, a decision tree is created by repeatedly dividing the data based on the values of different input features and by inducing a set of decision rules to predict clip labels based on the input features' values. The result is a tree-like data structure that has both internal nodes and leaf nodes, each associated with a decision rule, that collectively make up the decision criteria for classifying an input clip. As a final step during model creation, Gobert et al. (2013) reduced the set of input features by applying a standard feature selection technique called backward elimination search (Chandrashekar & Sahin, 2014).

The best-performing decision tree model that emerged from this process had, at its root, a decision rule about the number of adjacent controlled trials with repeats; if a student never ran two controlled experiments in a row, the model produced a high confidence prediction that the student did not know how to design a controlled experiment. If the student ran at least two controlled experiments in a row, the model utilized several additional features to determine how to classify knowledge of this skill. An example decision tree rule from Gobert et al. (2013) is shown in Box 7.1.

BOX 7.1: EXAMPLE DECISION RULE IN J48 DECISION TREE FOR DETECTING THE SKILL OF DESIGNING CONTROLLED EXPERIMENTS IN THE INQ-ITS TUTORING SYSTEM

IF count of adjacent controlled experiments (with repeats) = 1 AND
count of simulation variable changes ≤ 2 AND
count of pairwise controlled experiments (with repeats) > 1 AND
complete trials count > 2
THEN predict that the clip is a demonstration of designing controlled experiments with 74% confidence.

Source: Reproduced from Gobert et al. (2013)

After creating a domain model using supervised learning, as described above, the resulting model can be integrated back into the run-time AIED system. Specifically, the AIED system is extended to include support for distilling the predictor features utilized as input by the machine learning model (e.g., J48 decision tree) in real-time as students perform learning actions. The distilled features are assembled and provided as input to the trained model, which then produces a prediction, such as whether the student has demonstrated knowledge of how to design controlled experiments. These predictions can be used to drive decisions about adaptive support, problem selection, or other pedagogical functions of student modeling.

Another application of supervised learning for domain modeling in AIED systems is automated goal recognition, which is the task of inferring an agent's higher-order goal (or goals) based upon a series of observations of their lower-level actions in a learning environment (Sukthankar et al., 2014). A broad range of ML techniques have been used for automated goal recognition in AIED systems, including probabilistic models (Mott et al., 2006), deep-learning techniques (Min et al., 2016), and statistical relational learning methods that combine

ML with logic-based representations (Ha et al. 2011). Like the work by Gobert et al. (2013), ML-based goal recognition provides a framework for devising models that encode the relationships between low-level actions and higher-order cognitive processes, such as goal setting or strategy use.

Unsupervised learning

Unsupervised learning techniques are widely used to analyze student data in AIED systems. Unsupervised learning is a family of ML techniques for capturing generalizable patterns or groups in unlabeled datasets (Bishop, 2006). A key application of unsupervised learning is discovering patterns in student strategy use, an application that blends elements of domain modeling and student modeling. For example, Käser and Schwartz (2020) clustered students into different groups based on their inquiry strategies in an open-ended learning environment for middle-school physics education called TugLet. Although Käser and Schwartz (2020) used a rule-based model of knowledge of motion and forces in TugLet, they utilized clustering analysis to model inquiry strategies. Clustering analysis revealed several patterns of student behavior that could be interpreted in terms of effective and ineffective inquiry strategies in TugLet. For example, one cluster was associated with efficient, systematic testing behavior using TugLet's simulation-based exploration mode. Another cluster was associated with an inefficient trial-and-error process, which included little use of TugLet's exploration mode. In total, seven clusters were identified, and they were validated using a separate validation dataset collected from a different public middle school. Findings showed a high level of cluster agreement between the two datasets, although the distribution of clusters was different (Käser & Schwartz, 2020). Notably, the analysis also revealed a new, positive inquiry strategy—keep it simple to isolate equivalence of units—which had not yet been described in the literature.

Clustering analysis has also been used to identify student error patterns and misconceptions in specific domains. Shi and colleagues (2021) used student program code data from an introductory computing course to train a neural embedding-based representation of student code called code2vec. This representation was then used to identify clusters of student mistakes using Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996). The resulting clusters were interpreted by domain experts to identify common misconceptions exemplified in students' unsuccessful code. A case study of three clusters suggested that the approach revealed novel and meaningful misconceptions that would not have been easily discovered using traditional clustering methods (Shi et al., 2021).

Unsupervised learning techniques also play a role in acquiring domain knowledge representations for use by *simulated students*. For example, the SimStudent teachable agent, which is integrated with the APLUS learning by teaching platform, leverages a combination of inductive logic programming and learning by demonstration to learn rule-based domain model representations from demonstrations (e.g., feedback and hints) provided by ITS authors and students (Matsuda et al., 2020; Matsuda et al., 2013). SimStudent engages in a hybrid machine learning process that combines elements of both supervised and unsupervised learning. Demonstrations provided by students (a form of supervision) are used to generate production rules, each representing a skill, that together constitute SimStudent's model of the domain. Unsupervised representation learning algorithms that encode "deep features" in algebra equation solving have also been integrated with SimStudent, reducing the knowledge engineering requirements associated with constructing these types of agents (Li et al., 2015). Work on SimStudent has also been extended toward the creation of the Apprentice Learner

Architecture, which enables interactive machine-learning-based authoring paradigms for the creation of AIED systems across a range of knowledge types and domains (MacLellan & Koedinger, 2020; Weitekamp et al., 2020).

An important property of student behavior in many AIED systems is its sequential nature. Sequence mining has been found to be effective for detecting patterns in learner behavior that unfold over time within open-ended learning environments. Kinnebrew et al. (2013) developed a differential sequence mining framework to distinguish between productive and unproductive learning behaviors in Betty's Brain, a learning-by-teaching platform for middle-school mathematics and science education. They identified sequential patterns in students' metacognitive activities, such as taking quizzes or reading relevant resources to monitor aspects of their solutions, that were distinct between high- and low-performing groups of students, or between productive and unproductive phases of learning. Taub et al. (2018) used a similar approach to extract patterns in students' scientific reasoning processes (e.g., sequences of relevant and irrelevant testing behaviors) during inquiry-based learning in a game-based learning environment. A key use case for these patterns is to refine the design of metacognitive strategy feedback that is specific to a particular domain and learning environment. Other work has utilized principal component analysis (PCA) to filter students' problem-solving sequences to devise time series representations of learner behavior in open-ended learning environments (Reilly & Dede, 2019; Sawyer et al., 2018). Using PCA-based representations of expert problem solving, this approach enables entire trajectories of student problem-solving behaviors to be efficiently compared with expert problem-solving trajectories. The distance between student trajectories and expert trajectories has been found to be predictive of student learning gains (Sawyer et al., 2018). Again, these examples are illustrative of the close relationship between domain modeling and student modeling in many applications of ML (and data mining) within AIED.

Reinforcement learning

Recent years have seen growing interest in leveraging reinforcement learning to model and support students' learning processes (Doroudi et al., 2019). Reinforcement learning is a family of ML techniques that focus on sequential decision making under uncertainty (Sutton & Barto, 2018). Rather than being trained upon a set of labeled data, reinforcement learning typically involves a process of learning by experience where an ML agent explores alternative courses of action with the goal of maximizing the accumulated reward over time. Much of the work on reinforcement learning in AIED has focused on pedagogical models (Ausin et al., 2020; Beck et al., 2000; Rowe & Lester, 2015), but reinforcement learning techniques have also been used to model structured domains, such as logic proofs and algebra equation solving. Barnes and Stamper (2008) leveraged a Markov decision process (MDP) formalism to model logic proof solving in the DeepThought intelligent tutoring system. By mining data from previous students' interactions, their MDP-based domain representation could be used to automatically generate contextual hints by matching the current state of a student's proof attempt and selecting the next action that is optimal with respect to the MDP's reward function. The result was a system that would recommend solution paths (i.e., proof steps) that were taken in successful proofs by previous students. An additional benefit of this approach is that the hint generation model can continue to improve as new student data is collected; the domain model is enriched as more students interact with the system.

Rafferty and Griffiths (2015) utilized inverse reinforcement learning to interpret students' freeform algebra equation-solving steps to assess student knowledge. Their approach, which is based upon Bayesian inverse planning, enables the interpretation of students' equation-solving processes regardless of whether a structured or freeform interface is used by the learning environment. Rafferty and Griffiths (2015) modeled equation transformations as a Markov decision process, and they utilized inverse planning to approximate a posterior distribution over the space of hypotheses representing possible learner understandings in the algebra equation- solving domain. In related work, Rafferty et al. (2016) used inverse planning to provide automated feedback on students' equation-solving choices. Results from a study indicated that the feedback yielded pre-performance to post-performance improvements, and learners who received feedback on skills that were far from mastery showed greater improvement than students who received feedback on already-mastered skills.

How to create domain models using machine learning

There are software toolkits that are freely available to facilitate the creation of ML-based domain models, but current tools are not specific to domain model creation for use in AIED systems. Several popular tools provide graphical user interfaces for training, validating, and testing ML algorithms, such as RapidMiner (Kotu & Deshpande, 2014) and Weka (Hall et al., 2009). Similarly, research on deep learning applications has been accelerated by the availability of popular software packages for R (e.g., caret) and Python (e.g., TensorFlow, Keras, PyTorch, and ScikitLearn). A prerequisite for the application of ML techniques is the availability of cleaned, formatted data to train and validate models. Given the widespread availability of free ML toolkits, much of the work in creating ML-based models in AIED systems is in the acquisition and formatting of data, formulation of the ML task, and engineering of feature representations. Shared data repositories, such as DataShop and LearnSphere, provide datasets, analysis tools, and visualizations that can serve to reduce the burden of data acquisition and formatting (Koedinger et al., 2010; Koedinger et al., 2017). Furthermore, many ML toolkits provide implementations of common feature selection algorithms, which further reduce the burden of manually creating effective feature representations in ML-based domain models. Although these resources are useful, the development of authoring tools that specifically focus on the creation of ML-based domain models is a promising direction for future research.

Challenges

Machine learning provides a useful means for automatically modeling domain knowledge in AIED systems, but it also raises several important challenges. First, ML assumes the availability of significant data for training and validation. Sufficient data to produce a high-quality model may not always be available, or the available data may not be ideally distributed (e.g., a non-randomized problem order), especially early in the development of an AIED system or in domains that are not conducive to formalization using ML techniques. Second, ML approaches to domain modeling raise important questions related to fairness and transparency. Recent years have seen growing attention to the issue of encoded bias in ML (Gardner et al., 2019). In domain modeling, a biased model has the potential to cause harm to students if they are improperly judged to be engaged in "incorrect" learning behaviors because those behavior patterns were not included in the data used to train the model.

A related issue is transparency. Machine learning often produces models that are effectively a “black box” consisting of many thousands (or more) parameters that are difficult to interpret or explain. Lack of transparency, including weak interpretability and/or explainability, in a domain model is likely to reduce the level of trust that is imparted upon it, reduce what scientific insights can be gained from the model, and perhaps even reduce its utility for instruction. This issue points toward the opportunity for developing *explanatory models* within AIED systems, which provide insights about learners and the learning process that are interpretable and actionable in addition to being accurate (Rosé et al., 2019).

A third challenge is the issue of semantic model degeneracy, which refers to situations where the parameters induced for an ML-based domain model conflict with theoretical or conceptual understandings of the domain (Baker et al., 2008; Doroudi & Brunskill, 2017). This issue is closely related to concerns of model plausibility, identifiability, and consistency (Huang et al., 2015). An example occurs in work by Gobert et al. (2013), described above, in which they initially observed that ML-based detectors of students’ inquiry skills (e.g., designing controlled experiments) omitted key features considered theoretically important to the behavior, such as the number of controlled comparisons that a student made in his/her data set. This observation resulted in the adoption of a modified approach to their text replay tagging procedure and the use of different datasets for training, refining, and testing their machine learning model to improve its construct validity. This form of model degeneracy may reflect an issue in the distribution of the training data, or some other source of bias rather than properties of an ML model or an optimization algorithm. In general, however, ML-based models are more prone to degeneracy issues than the other major paradigms discussed in this chapter where domain experts are more heavily involved in the creation or refinement of domain models.

Despite these limitations, machine learning provides a range of useful tools and techniques for domain modeling in AIED systems. Furthermore, applications of ML intersect with many of the other paradigms outlined in this chapter, giving rise to hybrid systems that draw upon multiple paradigms to devise computational models of domain knowledge that inform the adaptive pedagogical functionalities of AIED systems.

Domain Modeling for Intelligent Textbooks

Domain modeling has supported a wide range of intelligent or adaptive online textbooks, which have a long history in personalized web-based learning (Brusilovsky & Pesin, 1998; Henze & Nejdl, 1999; Kavcic, 2004; Melis et al., 2001; Weber & Brusilovsky, 2001). Intelligent or adaptive textbooks can guide students to the most relevant content based on student modeling (Huang et al., 2016; Thaker et al., 2018), content recommendation (Kavcic, 2004; Sosnovsky et al., 2012), knowledge-adapted content presentation (Melis et al. 2001), and adaptive navigation support (Brusilovsky & Eklund, 1998; Brusilovsky & Pesin 1998; Henze & Nejdl, 1999; Weber & Brusilovsky, 2001), such as a “traffic light” approach to annotate links as content for which the student is deemed ready or not ready (Brusilovsky et al., 1996, 1998). A domain model in an intelligent textbook typically consists of a KC-to-item mapping which specifies domain concepts presented on a page or a section (i.e., an item), and sometimes also specifies the prerequisite concepts required to understand the current page or section. Domain models in intelligent textbooks are used mainly to assess student knowledge (Huang et al., 2016) and guide students to the right content (Brusilovsky & Eklund, 1998; Brusilovsky & Pesin 1998; Henze & Nejdl, 1999); a few also assess student work and provide problem-solving support

(Weber & Brusilovsky 2001). Many studies have confirmed the effectiveness of personalization approaches based on domain and student modeling for student learning with online textbooks (Brusilovsky & Eklund, 1998; Davidovic et al., 2003; Weber & Brusilovsky, 2001).

Domain modeling in intelligent textbooks has evolved from manual indexing by domain experts to the more recent automatic extraction using ML and text mining (i.e., natural language processing) techniques. Several studies demonstrated the effectiveness of personalization approaches built based on manually-indexed domain models in early adaptive textbooks (Brusilovsky & Eklund 1998; Davidovic et al., 2003; Weber & Brusilovsky 2001). To reduce the cost of expert labeling, a research stream focused on open corpus adaptive educational hypermedia (Brusilovsky & Henze, 2007) has explored automatic methods that borrow from information retrieval and semantic web models (Dolog & Nejdl, 2003; Sosnovsky & Dicheva, 2010) to build various kinds of models of educational documents. More recently, automatic methods have been developed for identifying prerequisite concepts (Agrawal et al., 2014; Labutov et al., 2017), building prerequisite structures (Chaplot et al., 2016), and building concept hierarchies (Wang et al., 2015). Chau et al. (2020) provide a comprehensive, offline evaluation of various automatic concept extraction methods and a review of domain modeling for adaptive textbooks.

A notable challenge is the evaluation of domain models used in intelligent textbooks. Early empirical evaluations in this field typically compared an adaptive system with a non-adaptive system, demonstrating benefits of a whole adaptivity "package" such as a combination of the domain model, student model, and adaptive navigation support (Brusilovsky & Eklund, 1998; Weber & Brusilovsky, 2001). There is still a lack of empirical or data-driven evaluations that isolate the contribution of domain models. Another challenge is a lack of high-quality labeled data to train and evaluate domain models for intelligent textbooks. Most automatic methods in this field rely on labeled data provided by domain experts for supervised or semi-supervised learning tasks, yet labeling domain concepts is a very time-consuming and difficult task.

Examples and Generalized Examples

Several types of AIED systems use problem-solving examples or generalized versions of such examples as their main representation of domain knowledge. These systems typically assess student work by comparing them against stored examples. To this end, they employ a flexible matching process for finding a relevant example and mapping it to the student solution. Given that there may be great variability in student solutions, matching literally against stored examples might not be effective.

Example-tracing tutors, a widely-used paradigm for creating AIED tutoring systems, use "behavior graphs" as their representation of domain knowledge (Aleven et al., 2016). Behavior graphs are generalized examples of step-by-step solution *processes* within a given problem. The nodes in these graphs represent problem states, the links represent problem-solving steps. Different paths in a behavior graph capture different ways of solving a given problem. (Thus, behavior graphs can handle problems with multiple different solution paths, although if the number of such paths is very large in a problem, they become unwieldy.) An example-tracing algorithm flexibly matches a student's problem-solving steps, one-by-one, against the graph, to track which path(s) the student might be following. The process of example tracing is analogous to model tracing, described in the current chapter, above. Just as the model-tracing algorithm tracks which solution path the student is following from among the many solutions paths the rule model could generate, the example-tracing algorithm tracks which solution path the student is on from among the multiple possible solution paths captured in the given behavior graph. With dedicated authoring tools, an author can create behavior graphs through programming by demonstration, without having to write code. With the same tools, the author can also indicate how a graph generalizes, so it can be used to recognize, as correct, a wide range of student problem-solving behavior. Many example-tracing tutors have been built and found to be effective in helping students learn in real educational settings (Aleven et al., 2016). Interestingly, behavior graphs have a long history in cognitive science (Newell & Simon, 1972) that predates their use in AIED systems, another way in which the fields of AIED and cognitive science are connected.

Other AIED systems use examples of *solutions* (rather than solution paths) as a key store of domain knowledge. We have already encountered one instance: constraint-based tutors use stored examples of solutions in order to help students learn. As another instance, a programming tutor by Rivers and Koedinger (2017) uses a large store of student Python programs to interpret student work and provide hints for how to fix or complete programs. Much emphasis in this work is on creating a process for converting students' program code to an abstract, canonical representation. Doing so decreases the number of examples that need to be stored and increases the probability of finding a match for any given student program.

In a different way of using examples, several AIED systems support students in studying worked examples as part of their instructional approaches, often eliciting self-explanations from students, with feedback from the system (see Conati & VanLehn, 2000; Adams et al., 2014, MacLaren et al., 2016; Zhi et al., 2019; Chen et al., 2020; also Chapter 9 by Aleven et al.). Some systems are capable of adaptively selecting or fading worked examples (i.e., gradually transitioning to problem solving), based on measures of students' knowledge or knowledge growth (Goguadze et al., 2011; Salden et al., 2010). In these methods for selecting or fading examples on an individualized basis, the domain model, student model, and pedagogical model all work together.

Knowledge Spaces

Another paradigm in domain modeling and student modeling is based on the theory of *knowledge spaces* (Doignon & Falmagne, 1985, 2012). In this paradigm, domain knowledge is represented by a network of interconnected items. These items correspond to the problem types in the given educational domain. The relations between items are precedence relations between problem types, which may be due to the prerequisite structure of the domain or the order in which problem types are taught (Falmagne et al., 2006). A student's knowledge state is represented as a subset of these items, namely, the items that the student is assumed to have mastered. The knowledge spaces paradigm could be viewed as overlapping with the Bayesian network paradigm in that a knowledge structure in knowledge spaces theory can be described as a Bayesian network without hidden nodes, where each of the nodes maps to a concrete class of problems (Desmarais & Pu, 2005). This paradigm emphasizes tailoring and selecting the right learning content based on estimated student competence.

Systems based on the knowledge spaces theory include the widely used ALEKS mathematics tutor (Falmagne et al., 2006) and the Catalyst or MCWeb system for chemistry (Arasasingham et al., 2011, 2005). A recent meta-analysis (Fang et al., 2019) revealed that ALEKS was as effective as, but not better than, traditional classroom teaching. Also, a study by Arasasingham et al. (2005) found that the MCWeb system improved learning outcomes in chemistry. Students who used MCWeb for their homework performed significantly better in subsequent assessments, compared with students who carried out homework from their textbooks. However, it's unclear whether the improvement was from the multiple representation mechanism used in the system, or the knowledge spaces model itself.

Knowledge spaces are often human engineered, but they need to be refined with data to enhance accuracy (Falmagne et al., 2006; Pavlik et al., 2013). Typically, a large amount of data is needed to infer a complete set of precedence relations. Partial Order Knowledge Structures (POKS) have been developed to address challenges of inferring the AND/OR precedence relations from data alone (Desmarais et al., 2006; Desmarais & Pu, 2005). Another issue is that this paradigm does not model domain knowledge in terms of cognitive structures (e.g., concepts, skills). Without such information, it is hard to differentiate problems that share some knowledge demands but differ in other knowledge demands, which would seem to hamper accurate knowledge estimation or recommendations. There is some work to extend the framework to include skills as well (Heller et al., 2006). For a more thorough review of this paradigm, readers can refer to reviews by Desmarais and Baker (2012) or by Pavlik et al. (2013).

Other Types of Domain Models

We briefly discuss several other types of domain models that have been demonstrated to be effective in AIED systems, even if they have not quite seen the same amount of research or widespread use as the paradigms discussed above: domain ontologies, concept maps, case-based models, qualitative models of physical systems, models of inquiry processes, and issue-based models

First, a substantial amount of work within the field of AIED has focused on using *domain ontologies* within AIED systems (see Lenat & Durlach, 2014). These ontologies provide a hierarchical ordering of domain concepts. For example, the SlideTutor system uses domain knowledge represented as an ontology. This representation is used for visual classification problem solving in surgical pathology, in combination with a rule-based model (Crowley et al., 2003; Crowley & Medvedeva, 2006). Separating declarative knowledge (represented in the system's ontology) and procedural knowledge (represented as rules) facilitates system extensions (e.g., modeling additional medical visual classification tasks) and reusability of knowledge representations. Ontologies have also been used for domain model alignment between various AIED systems to enable the exchange of student models. For example, Sosnovsky et al. (2009) discuss how AIED systems based on different formalisms can exchange assessments of student work using ontology mapping. The ontology served as a mediator between the set of constraints in SQL-Tutor and SQL-Guide, an adaptive hypermedia system in which the student model is represented as an overlay on the ontology.

Second, a number of AIED systems use *concept maps* (Novak, 1996) and *conceptual graphs* (Sowa, 1994) as domain models. Concept maps are graphs that represent domain concepts and relations between them, with domain-specific labels, to capture conceptual and propositional knowledge of a domain (Martínez-Maldonado et al., 2010). In a number of AIED systems, the students' main learning activity is to construct a concept map as a representation of his/her domain knowledge. They may receive assistance from the system (e.g., the Betty's Brain system from Biswas et al., 2005) or from a teacher, in turn supported by an analytics system (e.g., Martinez-Maldonado et al., 2014). Another approach involves the student in building a student model, represented as a conceptual graph, collaboratively with the STyLE-OLM system (Dimitrova, 2003; Dimitrova & Brna, 2016), to improve the accuracy of the student model, and at the same time promote metacognitive skills such as reflection and self-assessment.

Concept maps have also been used to provide some guidance to students during inquiry activities (Hagemans et al., 2013). All these approaches capitalize on the idea that concept maps or conceptual graphs are highly interpretable ways of organizing domain knowledge. The system often assesses the student's work by comparing the student's evolving model against a stored expert model but may also assess student work in terms of whether their process of building and testing the model, given resources, is sound and coherent.

Third, some AIED systems have *case-based domain models*, which capture processes of reasoning with cases. Some AIED systems support students in domains where experts naturally reason with cases. In such domains, rules may be "weak" or too abstract to give much direct guidance (i.e., these may be ill-defined domains); therefore, reasoning with cases is a natural supplement or even a substitute for reasoning with rules (Lynch et al., 2009). Examples of such systems are CATO (Aleven, 2003) and LARGO (Pinkwart et al., 2009) in the domain of legal reasoning. Some systems use case-based reasoning processes as part of their pedagogical approach (Gauthier et al., 2007; Schank, 1990), although that is outside the scope of the current chapter.

Fourth, some AIED work has focused on systems that use various types of "qualitative models" as domain models (Bredeweg & Forbus, 2016; Bredeweg et al., 2013; Joyner et al., 2013). Betty's Brain, mentioned above, could also be viewed as falling into this category (Biswas et al., 2016). Typically, these models capture phenomena in the natural world (e.g., models of biological or ecological systems, models of how physical devices work, or other forms of causal models). They are networks that capture causal influences among variables; unlike concept maps or conceptual graphs, discussed above, these networks can support inference generation. Oftentimes, they are used in AIED systems in which the student's task is to build a model (e.g., of a natural phenomenon), with guidance from the system. In some systems, the qualitative models are executable, so that students can test their models against data (e.g., to see if they account for data about the model phenomenon).

Fifth, relatedly, some AIED systems use *models of inquiry processes* to support students as they carry out such processes, such as the Inq-ITS system from Gobert et al. (2013) and Käser & Schwartz's (2020) work with the TugLet game. These models capture how to gather evidence and design experiments and are used to assess student work and knowledge of inquiry skills; the models used in Inq-ITS are described above, in the section in this chapter on machine learning. In some instances, models of inquiry processes have been found to generalize across different science topics (Gobert et al., 2013). More information about the nature of these systems and their domain models can be found in Chapter 9 by Aleven et al.

Finally, some systems use an *issue-based approach*. They use a domain model that captures specific "issues" within the given task domain, which include both desirable aspects and shortcomings of work that are common among novices. Having a representation of issues enables the system to provide useful adaptive feedback on student work, without the need for a complete domain model. Systems of this kind are often characterized by a coaching style that minimizes the amount of intervention (Burton & Brown, 1982). An issue-based approach may be particularly appropriate in domains where AI models have limited scope or are not sophisticated enough to fully evaluate students' work. The LARGO system of Pinkwart et al. (2009) illustrates this approach. Using LARGO, students diagram out oral argument exchanges before the US Supreme Court. The LARGO system provides feedback based on a representation of argument issues that were not themselves meant to be a complete model of the kind of legal argumentation being analyzed. As an additional example, some analytics-based tools that provide alerts regarding student work to teachers (e.g., VanLehn et al., 2021) can also be viewed as falling under this category.

New Trends

We briefly review two lines of research related to domain modeling that have come to the foreground since 2015: the application of text mining techniques for discovering and refining domain models from text and the use of crowdsourcing for building domain models.

A new trend in AIED research is discovering and refining domain models from text utilizing text mining techniques. The availability of large amounts of digital text-based learning content and advances in computational linguistics create numerous opportunities for automated discovery or improvement of domain models (for example, a Q-matrix or prerequisite structures) based on text mining. This trend goes beyond the textbook context mentioned in our previous section and involves broader contexts such as Wikipedia (Gasparetti et al., 2015), MOOC lecture transcripts (Alsaad et al., 2018), and conversational interactions with learners (see Chapter 11 by Rus et al. for text-based domain modeling in AutoTutor systems). Some of the approaches used only textual content (Alsaad et al., 2018; Gasparetti et al., 2015; Pardos & Dadu, 2017), while others used a combination of textual and student performance data (Chaplot et al., 2016; Chen et al., 2018; Matsuda et al., 2015). Various text mining techniques have been shown to be effective, as evaluated by predictive performance in prediction tasks. For example, Pardos and Dadu (2017) applied a skip-gram model (a neural network commonly known as word2vec) to model both the content of a problem and problems around it and reached a 90% accuracy in predicting the missing skill from a KC model. Michalenko et al. (2017) utilized word embeddings to detect misconceptions from students' textual responses to open-response questions. Other text mining techniques have also been applied, such as semantic analysis for identifying relevant Wikipedia concepts in text-based learning objects (Gasparetti et al., 2015), named entity recognition for extracting educational concepts in curriculum standards (Chen et al., 2018), and correlational topic modeling for identifying topics by analyzing the content of mathematics problems (Slater et al., 2017).

Another new trend that has received increasing attention in recent years is crowdsourcing domain models. This work is situated in the broader ongoing research into crowdsourcing explanations, feedback, and other pedagogical interactions (Heffernan et al., 2016; Williams et al., 2016). For example, crowdsourcing has been used to identify KCs in the domains of mathematics and English writing; it was found that roughly one-third of the crowdsourced KCs directly matched those generated by domain experts (Moore et al., 2020). *Learnersourcing* is a form of crowdsourcing where learners collectively contribute novel content for future learners while they are engaged in a meaningful learning experience themselves (Kim, 2015). One active line of learningsourcing research is in video learning, including subgoal label generation (Weir et al., 2015), solution structure extraction (Kim, 2015), and concept map generation (Liu et al., 2018) from educational videos. Several studies found that learner-generated labels or content can be comparable in quality to expert-generated ones (Liu et al., 2018; Weir et al., 2015), and that the learnersourcing workflow did not detract learners from the learning experience (Weir et al., 2015). Learnersourcing was also used for understanding large-scale variation in student solutions in programming and hardware design, and has proven valuable for both teachers and students (Glassman & Miller, 2016).

DISCUSSION

Our review of AIED domain modeling paradigms, summarized in Tables 7.1 and 7.2, shows an astounding richness. Many domain modeling paradigms for AIED systems have been explored, developed, and proven to be useful. Within each paradigm, multiple systems have been built and have been found to be effective in helping students learn in real educational settings.

The use of domain models in AIED systems offers many advantages, as described above and summarized in Table 7.1. A domain model enables a system to provide adaptive step-level guidance to students within complex problem-solving activities, which enables the system to support student learning more effectively than systems without step-level support (VanLehn, 2011) and to guide students during richer problem-solving experiences. Domain models can be the basis for student modeling and hence for many forms of adaptivity within AIED systems, including personalized mastery learning. As well, domain models can be used to guide the design of many aspects of AIED systems. Finally, some have argued that the use of domain models may promote interoperability between system components and systems (Sottilare et al., 2016, Chapter 4). Although domain models are key in many effective AIED systems, it should be noted that several adaptive instructional systems, including some widely used ones, do not have a strong domain model, for example ASSISTments (Heffernan & Heffernan, 2014), MathSprings (Arroyo et al., 2014), Khan Academy (Kelly & Rutherford, 2017), and Duolingo (von Ahn, 2013). Without domain models, these systems provide simple practice problems with immediate feedback and forms of personalized task selection, which can be effective for learners. These systems, however, cannot support the greater adaptivity and more complex problem solving afforded by a domain model.

Although, as Table 7.1 attests, there is substantial overlap in the functionality supported by the different domain modeling paradigms, some interesting differences emerge as well. Such differentiation across AIED paradigms is attractive, for example because it gives the AIED system developer different tools in their toolkit. For example, the AIED domain modeling paradigms differ in the degree to which they can readily handle problems that have large solution spaces. In such problems (e.g., computer programming or solving algebraic equations), there is great variability in student solutions, even among the correct solutions, or the steps that lead to correct solutions. Constraint-based tutors and model-tracing tutors deal well with such domains. On the other hand, example-tracing tutors are not well-suited to practice problems with large solution spaces, although they can still handle problems with multiple alternative solution paths. In addition, although BNs in theory could be constructed to support domain modeling with vast solution spaces, the construction of such BNs may be labor intensive if done by human engineering, or challenging in terms of both computational efficiency and accuracy if done by automated methods. Systems that rely on stores of examples have the potential to handle large solution spaces, although doing so might require sophisticated canonicalization and matching approaches (e.g., Rivers & Koedinger, 2017). Advances in machine learning have shown promise to tackle some of these challenges.

Different domain modeling paradigms assess student work in different ways. In some paradigms (e.g., model-tracing tutors, BNs in Andes), the domain model captures general problem-solving knowledge, which enables the system to simulate the process of solving problems and to assess student solutions steps by comparing them against solution steps or processes generated by the model. In other paradigms (e.g., constraint-based tutors), the domain model

Table 7.2 *Strengths and weaknesses of AIED domain modeling paradigms*

Paradigms	Can handle large solution spaces	Amenable to machine learning or data-driven refinement	Ease of authoring/authoring tools exist	Requiring data to develop	Model identifiability and/or degeneracy issues	Interpretability
Rules	Yes	Yes, SimStudent can learn rules. Many methods for data-driven KC model refinement	CTAT, SimStudent	Cognitive task analysis upfront recommended	N/A	Yes
Constraints	Yes	Yes	Requires domain knowledge/ ASPIRE	No data required (though can use data if you have it)	N/A	Yes
Behavior Graphs	No	Many methods for data-driven KC model refinement.	CTAT—non-programmer authoring	Cognitive task analysis upfront recommended	N/A	Yes
Bayesian Networks	Hard	Yes	Yes, but not integrated with ITS authoring	Yes	Yes	Yes
Supervised Learning	Hard	—	Yes, but not integrated with ITS authoring	Yes	Yes	Depends
Unsupervised Learning	Yes	—	Yes, but not integrated with ITS authoring	Yes	No. No conceptual assumptions.	Depends
Reinforcement Learning	Yes	—	Yes, but not integrated with ITS authoring	Yes	In principle, yes.	Depends

captures knowledge for assessing solutions and partial solutions. This approach enables the system to provide rich feedback on student work without the ability to generate problem solutions. Issue-based approaches capture knowledge for evaluating specific solution aspects (e.g., identifying inquiry strategies or recognizing common issues or undesirable properties of student work) without fully evaluating a complete solution (issue-based approaches or some ML approaches). Finally, some example-based approaches store solutions or solution paths, and have smart, flexible ways of comparing student solutions to stored solutions, so as to provide feedback and hints (Aleven et al., 2016; Rivers & Koedinger, 2017; Stamper et al., 2013).

Relatedly, across domain models, we see some differences in the type of formative feedback that the systems can give to students. As mentioned, formative feedback is a key function of domain models. Many domain modeling approaches evaluate the overall quality of the student's work (e.g., correctness of the entire solution), but others (e.g., issue-based approaches and some ML approaches) focus only on specific aspects of solutions, without the intent of assessing the overall quality of student work. An issue-based approach may be appropriate in domains where AI/ML has not advanced to the point that complete evaluation of solution quality is possible (e.g., complex, open-ended domains; Lynch et al., 2009). Another difference in tutoring behaviors is in the specificity of next-step hints. Model-tracing tutors and example-tracing tutors can always suggest a specific next step (together with reasons why). On the other hand, constraint-based tutors provide hints that state missing solution elements, or extra elements; they can also present example solutions to students. A final observation is that most domain modeling paradigms allow for modeling of errors or misconceptions for the purpose of providing error-specific feedback to students, though without strictly requiring it.

Although, as mentioned, creating a domain model is labor intensive, various developments make doing so easier, including the development of AIED authoring tools, the vastly increased use of ML in creating AIED systems, and (in a very recent development) the use of crowdsourcing (Williams et al., 2016; Yang et al., 2021). AIED authoring tools have long existed for established domain modeling paradigms such as rules (Aleven et al., 2006b; Koedinger et al., 2003), constraints (Mitrovic et al., 2009; Suraweera et al., 2010), and behavior graphs (Aleven et al., 2016). A number of these projects involve machine learning to support non-programmers in creating domain models for use in AIED systems (Matsuda et al., 2015; Mitrovic et al., 2009), often integrated with other authoring functions (e.g., tutor interfaces, student models, etc.). We see an on-going trend toward the use of ML to facilitate representation learning (Li et al., 2018), factoring interactive learning approaches to domain knowledge (MacLellan & Koedinger, 2020), and addressing human–computer interaction aspects (Weitekamp et al., 2020). We also see opportunities for further integration, both between AIED domain modeling paradigms and with existing general AI/ML toolkits, to facilitate experiments with AIED systems.

The use of machine learning to create domain models has grown dramatically in recent years. Machine learning has been used to create domain models that range in focus from specific aspects of student work (e.g., Gobert et al., 2013) to more general models of problem-solving knowledge within a specific domain (e.g., Barnes & Stamper, 2008). A broad range of machine learning techniques have also been used across different domain modeling tasks, including supervised learning, unsupervised learning, and reinforcement learning methods. Interestingly, machine learning is not tied to a particular computational representation or formalism. Rather, machine learning can be used to create domain models (and student models) with a broad range of representations, including the major paradigms discussed in this chapter (e.g., rules, constraints, Bayesian networks, etc.) as well as others. In fact, data-driven techniques for creating or refining domain models, which are informed by machine learning, have proven useful across all domain modeling paradigms (see Table 7.2). The data-driven methods used in AIED research differ starkly with respect to the amount of data needed. Some paradigms depend on the availability of large amounts of data right from the start (i.e., many ML approaches). Others can operate while requiring less or even no data initially (e.g., interactive ML or qualitative cognitive task analysis; Clark et al., 2007). Some require limited data for the initial creation of a model, but require more data later (e.g., for data-driven refinement). We see great promise for approaches and tools that integrate data-driven improvement and authoring of AIED systems, a challenge (and an opportunity!) for the field.

The review highlights many connections and dependencies between an AIED system's domain model and its student model, traditionally considered to be separate modules. These two models tend to be closely coupled and sometimes fully integrated. For example, overlay models and KC modeling are ways of tightly linking domain models and student models. As another example, BNs have been applied in an AIED system (VanLehn et al., 2005) in a manner that elegantly blends domain modeling with student modeling. Another interesting connection is seen where ML-based domain models are trained from student learning data, or when a KC model is refined using log data from a tutoring system.

An interesting issue arises regarding the *practice* of AIED: given that professional AIED designers, developers, and learning engineers have multiple domain modeling paradigms to choose from, how should they select the most appropriate paradigm for any given AIED development project or the most appropriate combination of paradigms (see Roll et al., 2010)? So far, the field of AIED has not produced a strong, evidence-based set of guidelines; generating such guidelines is a great challenge because research studies comparing different domain modeling paradigms are very hard to do. Based on our review, some relevant factors that influence this choice may be:

1) The desired behavior of the tutoring system. We noted some differences with respect to whether feedback is based on a full evaluation of correctness or focuses on specific issues only, the specificity of next-step advice, and whether error feedback is accommodated.
2) Whether the tutored tasks have large solution spaces. We noted some differences among the domain modeling paradigms in how well they deal with large solution spaces.
3) Whether one has confidence that AI can do a good enough job either in generating or fully evaluating solutions. We noted some differences among the paradigms in this regard as well. This factor may correlate with how well- or ill-defined the domain is.
4) The skill and experience of the design and development team (e.g., whether they have experience with the given modeling paradigm).
5) The availability of dedicated authoring tools or the demonstrated use of ML techniques to facilitate building the system.

To conclude, domain models are a key feature of many AIED systems. They support many of the behaviors that distinguish AIED systems from other educational technologies. AIED systems use a very wide range of AI knowledge representations. We hope that this chapter succeeds in highlighting that great richness, as well as the many advantages that derive from having a domain model.

REFERENCES

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, *6*, 52138–52160.

Adams, D. M., McLaren, B. M., Durkin, K., Mayer, R. E., Rittle-Johnson, B., Isotani, S., & van Velsen, M. (2014). Using erroneous examples to improve mathematics learning with a web-based tutoring system. *Computers in Human Behavior*, *36*, 401–411.

Agrawal, R., Gollapudi, S., Kannan, A., & Kenthapadi, K. (2014). Study Navigator: An algorithmically generated aid for learning from electronic textbooks. *Journal of Educational Data Mining*, *6*(1), 53–75.

Aleven, V. (2003). Using background knowledge in case-based legal reasoning: A computational model and an intelligent learning environment. *Artificial Intelligence*, *150*(1–2), 183–237.

Aleven, V. (2010). Rule-based cognitive modeling for intelligent tutoring systems. In R. Nkambou, J. Bourdeau, & R. Mizoguchi (Eds.), *Advances in intelligent tutoring systems* (pp. 33–62). Berlin, Germany: Springer.

Aleven, V., Connolly, H., Popescu, O., Marks, J., Lamnina, M., & Chase, C. (2017). An adaptive coach for invention activities. *International conference on artificial intelligence in education* (pp. 3–14). Cham, Switzerland: Springer.

Aleven, V., & Koedinger, K. R. (2013). Knowledge component (KC) approaches to learner modeling. *Design Recommendations for Intelligent Tutoring Systems*, *1*, 165–182.

Aleven, V., McLaren, B., Roll, I., & Koedinger, K. (2006a). Toward meta-cognitive tutoring: A model of help seeking with a Cognitive Tutor. *International Journal of Artificial Intelligence in Education*, *16*(2), 101–128.

Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. R. (2006b). The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. *International conference on intelligent tutoring systems* (pp. 61–70). Berlin, Heidelberg, Germany: Springer.

Aleven, V., McLaren, B. M., Sewall, J., van Velsen, M., Popescu, O., Demi, S., … Koedinger, K. R. (2016). Example-tracing tutors: Intelligent tutor development for non-programmers. *International Journal of Artificial Intelligence in Education*, *26*(1), 224–269. doi:10.1007/s40593-015-0088-2.

Alsaad, F., Boughoula, A., Geigle, C., Sundaram, H., & Zhai, C. (2018). Mining MOOC lecture transcripts to construct concept dependency graphs. *Proceedings of the 11th International Conference on Educational Data Mining* (pp. 467–473).

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the LISP tutor. *Cognitive Science*, *13*(4), 467–505.

Anderson, J. R., Conrad, F. G., Corbett, A.T., Fincham, J.M., Hoffman, D., & Wu, Q. (1993). Computer programming and transfer. In J. R. Anderson (Ed.), *Rules of the mind* (pp. 205–233). Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, *4*(2), 167–207.

Arasasingham, R. D., Martorell, I., & McIntire, T. M. (2011). Online homework and student achievement in a large enrollment introductory science course. *Journal of College Science Teaching*, *40*, 70–79.

Arasasingham, R. D., Taagepera, M., Potter, F., Martorell, I., & Lonjers, S. (2005). Assessing the effect of web-based learning tools on student understanding of stoichiometry using knowledge space theory. *Journal of Chemical Education*, *82*, 1251–1262.

Arroyo, I., Woolf, B. P., Burelson, W., Muldner, K., Rai, D., & Tai, M. (2014). A multimedia adaptive tutoring system for mathematics that addresses cognition, metacognition and affect. *International Journal of Artificial Intelligence in Education*, *24*(4), 387–426.

Ausin, M. S., Maniktala, M., Barnes, T., & Chi, M. (2020). Exploring the impact of simple explanations and agency on batch deep reinforcement learning induced pedagogical policies. In I. I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin & E. Millán (Eds.), *Proceedings of the International Conference on Artificial Intelligence in Education* (pp. 472–485). Cham, Switzerland: Springer.

Baghaei, N., Mitrovic, A., & Irwin, W. (2007). Supporting collaborative learning and problem-solving in a constraint-based CSCL environment for UML class diagrams. *International Journal of Computer-Supported Collaborative Learning*, *2*(2), 159–190.

Baker, R., Corbett, A., & Aleven, V. (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian knowledge tracing. *International Conference on Intelligent Tutoring Systems* (pp. 406–415). Berlin, Heidelberg, Germany: Springer.
Baker, R., Corbett, A. T., & Koedinger, K. R. (2007). The difficulty factors approach to the design of lessons in intelligent tutor curricula. *International Journal of Artificial Intelligence in Education*, *17*(4), 341–369.
Baker, R., Mitrovic, A., & Mathews, M. (2010). Detecting gaming the system in constraint-based tutors. In de Bra, P., Kobsa, A., Chin, D. (Eds.), *Proceedings of the 18th International Conference on User Modeling, Adaptation, and Personalization* (pp. 267–278). Berlin, Heidelberg, Germany: Springer.
Barnes, T., & Stamper, J. (2008). Toward automatic hint generation for logic proof tutoring using historical student data. In Woolf, B., Aïmeur, E., Nkambou, R., Lajoie, S. (Eds.), *Proceedings of the 9th International Conference on Intelligent Tutoring Systems* (pp. 373–382). Berlin, Heidelberg, Germany: Springer.
Beck, J. E., & Chang, K. M. (2007). Identifiability: A fundamental problem of student modeling. *International Conference on User Modeling* (pp. 137–146). Berlin, Heidelberg, Germany: Springer.
Beck, J., Woolf, B. P., & Beal, C. R. (2000). ADVISOR: A machine learning architecture for intelligent tutor construction. In H. Kautz & B. Porter (Eds.), *Proceedings of the 17th National Conference on Artificial Intelligence* (pp. 552–557). Menlo Park, CA: AAAI.
Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.
Biswas, G., Leelawong, K., Schwartz, D., Vye, N., & The Teachable Agents Group at Vanderbilt. (2005). Learning by teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence*, *19*(3–4), 363–392.
Biswas, G., Segedy, J. R., & Bunchongchit, K. (2016). From design to implementation to practice a learning by teaching system: Betty's Brain. *International Journal of Artificial Intelligence in Education*, *26*(1), 350–364.
Boroš, P., Nižnan, J., Pelánek, R., & Řihák, J. (2013). Automatic detection of concepts from problem solving times. *Proceedings of 16th International Conference on Artificial Intelligence in Education* (pp. 595–598).
Bredeweg, B., & Forbus, K. D. (2016). Qualitative representations for education. In R. A. Sottilare et al. (Eds.), *Design recommendations for intelligent tutoring systems: Volume 4—Domain modeling* (pp. 57–68). Orlando, FL: US Army Research Laboratory.
Bredeweg, B., Liem, J., Beek, W., Linnebank, F., Gracia, J., Lozano, E., ... & Mioduser, D. (2013). DynaLearn–An intelligent learning environment for learning conceptual knowledge. *AI Magazine*, *34*(4), 46–65.
Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In R. R. Burton, J. S. Brown & D. Sleeman (Eds.), *Intelligent Tutoring Systems* (pp. 79–98). New York: Academic Press.
Brownston, L., Farrell, R., Kant, E., & Martin, N. (1985). *Programming expert systems in OPS5: An introduction to rule-based programming*. Addison-Wesley Longman.
Brusilovsky, P., & Eklund, J. (1998). A study of user model based link annotation in educational hypermedia. *Journal of Universal Computer Science*, *4*(4), 429–448.
Brusilovsky, P., & Henze, N. (2007). Open corpus adaptive educational hypermedia. *The adaptive web* (pp. 671–696). Berlin, Heidelberg, Germany: Springer.
Brusilovsky, P., & Pesin, L. (1998). Adaptive navigation support in educational hypermedia: An evaluation of the ISIS-Tutor. *Journal of Computing and Information Technology*, *6*(1), 27–38.
Brusilovsky, P., Schwarz, E., & Weber, G. (1996). ELM-ART: An intelligent tutoring system on the world-wide web. In C. Frasson, G. Gauthier, & A. Lesgold (Eds.), *The Third International Conference on Intelligent Tutoring Systems* (pp. 261–269). Springer.
Burns, H., & Capps, C. (1988). Foundations of Intelligent Tutoring Systems: An introduction. In M. Polson & J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems* (pp. 1–19). Hillsdale, NJ: Lawrence Erlbaum Associates.
Carmona, C., Millán, E., Pérez-de-la-Cruz, J. L., Trella, M., & Conejo, R. (2005). Introducing prerequisite relations in a multi-layered Bayesian student model. *International Conference on User Modeling* (pp. 347–356). Berlin, Heidelberg, Germany: Springer.

Cen, H., Koedinger, K., & Junker, B. (2006). Learning Factors Analysis–A general method for cognitive model evaluation and improvement. *Proceedings of the 8th International Conference on Intelligent Tutoring Systems* (pp. 164–175). Berlin, Heidelberg, Germany: Springer.

Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16–28.

Chaplot, D. S., MacLellan, C., Salakhutdinov, R., & Koedinger, K. (2018). Learning cognitive models using neural networks. *Proceedings of the 19th International Conference on Artificial Intelligence in Education* (pp. 43–56). Cham, Switzerland: Springer.

Chaplot, D. S., Yang, Y., Carbonell, J., & Koedinger, K. R. (2016). Data-driven automated induction of prerequisite structure graphs. *Proceedings of the 9th International Conference on Educational Data Mining* (pp. 318–323). Raleigh, NC.

Chau, H., Labutov, I., Thaker, K., He, D., & Brusilovsky, P. (2020). Automatic concept extraction for domain and student modeling in adaptive textbooks. *International Journal of Artificial Intelligence in Education*, 31(4), 820–846.

Chen, Y., González-Brenes, J. P., & Tian, J. (2016). Joint discovery of skill prerequisite graphs and student models. *Proceedings of the 9th International Conference on Educational Data Mining* (pp. 46–53).

Chen, P., Lu, Y., Zheng, V. W., Chen, X., & Li, X. (2018). An automatic knowledge graph construction system for K-12 education. *Proceedings of the Fifth Annual ACM Conference on Learning at Scale* (pp. 1–4). London, United Kingdom.

Chen, X., Mitrovic, A., & Mathews, M. (2020). Learning from worked examples, erroneous examples, and problem solving: Toward adaptive selection of learning activities. *IEEE Transactions on Learning Technologies*, *13*(1), 135–149.

Clark, R. E., Feldon, D., van Merriënboer, J., Yates, K., & Early, S. (2007). Cognitive task analysis. In J. M. Spector, M. D. Merrill, J. J. G. van Merriënboer & M. P. Driscoll (Eds.), *Handbook of research on educational communications and technology* (3rd ed., pp. 577–593). Lawrence Erlbaum Associates.

Collins, J., Greer, J., & Huang, S. (1996). Adaptive assessment using granularity hierarchies and Bayesian Nets. In C. Frasson, G. Gauthier & A. Lesgold (Eds.), *Proceedings of the 3rd International Conference on Intelligent Tutoring Systems* (pp. 569–577). Springer-Verlag.

Conati, C., Gertner, A. & VanLehn, K. (2002). Using Bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction*, *12*(4), 371–417.

Conati, C., & VanLehn, K. (2000). Toward computer-based support of meta-cognitive skills: A computational framework to coach self-explanation. *International Journal of Artificial Intelligence in Education*, *11*, 389–415.

Corbett, A., & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, *4*(4), 253–278.

Corbett, A., Kauffman, L., MacLaren, B., Wagner, A., & Jones, E. (2010). A cognitive tutor for genetics problem solving: Learning gains and student modeling. *Journal of Educational Computing Research*, *42*(2), 219–239.

Corbett, A., McLaughlin, M., & Scarpinatto, K. C. (2000). Modeling student knowledge: Cognitive tutors in high school and college. *User Modeling and User-Adapted Interaction*, *10*(2), 81–108.

Crowley, R., & Medvedeva, O. (2006). An intelligent tutoring system for visual classification problem solving. *Artificial Intelligence in Medicine*, *36*(1), 85–117.

Crowley, R., Medvedeva, O., & Jukic, D. (2003). SlideTutor: A model-tracing intelligent tutoring system for teaching microscopic diagnosis. *Proceedings of the 11th International Conference on Artificial Intelligence in Education* (pp. 157–164).

Davidovic, A., Warren, J., & Trichina, E. (2003). Learning benefits of structural example-based adaptive tutoring systems. *IEEE Transactions on Education*, *46*(2), 241–251.

Davis, R., & King, J. J. (1984). The origin of rule-based systems in AI. In B. G. Buchanan & E. H. Shortliffe (Eds.), *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project* (pp. 20–51). Boston, MA: Addison-Wesley.

Desmarais, M. C., & Baker, R. S. (2012). A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction*, *22*(1–2), 9–38.

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

Desmarais, M. C., & Naceur, R. (2013). A matrix factorization method for mapping items to skills and for enhancing expert-based Q-matrices. *Proceedings of the 16th International Conference on Artificial Intelligence in Education* (pp. 441–450). Berlin, Heidelberg, Germany: Springer.

Desmarais, M. C. & Pu, X. (2005). A Bayesian student model without hidden nodes and its comparison with Item Response Theory. *International Journal of Artificial Intelligence in Education*, *15*, 291–323.

Dimitrova, V. (2003). STyLE-OLM: interactive open learner modelling. *International Journal of Artificial Intelligence in Education*, *13*, 35–78.

Dimitrova, V., & Brna, P. (2016). From interactive open learner modelling to intelligent mentoring: STyLE-OLM and beyond. *International Journal of Artificial Intelligence in Education*, *26*(1), 332–349.

Doignon, J. P., & Falmagne, J. C. (1985). Spaces for the assessment of knowledge. *International Journal of Man-Machine Studies*, *23*(2), 175–196.

Doignon, J. P., & Falmagne, J. C. (2012). *Knowledge spaces*. Springer Science & Business Media.

Dolog, P., & Nejdl, W. (2003). Challenges and benefits of the semantic web for user modelling. *Proceedings of the Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2003) at 12th International World Wide Web Conference*. Budapest, Hungary.

Doroudi, S., Aleven, V., & Brunskill, E. (2019). Where's the reward? A review of reinforcement learning for instructional sequencing. *International Journal of Artificial Intelligence in Education*, *29*(4), 568–620.

Doroudi, S., & Brunskill, E. (2017). The misidentified identifiability problem of Bayesian knowledge tracing. In X. Hu, T. Barnes, A. Hershkovitz & L. Paquette (Eds.), *Proceedings of the 10th International Conference on Educational Data Mining* (pp. 143–149).

Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han & U. Fayyad (Eds.), *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* (pp. 226–231). Palo Alto, CA: AAAI.

Falmagne, J. C., Cosyn, E., Doignon, J. P., & Thiéry, N. (2006). The assessment of knowledge, in theory and in practice. In R. Missaoui & J. Schmid (Eds.), *International Conference on Formal Concept Analysis (ICFCA), Lecture Notes in Computer Science* (pp. 61–79). Berlin, Heidelberg, Germany: Springer.

Fang, Y., Ren, Z., Hu, X., & Graesser, A. C. (2019). A meta-analysis of the effectiveness of ALEKS on learning. *Educational Psychology*, *39*(10), 1278–1292.

Ganeshan, R., Johnson, W. L., Shaw, E., & Wood, B. P. (2000). Tutoring diagnostic problem solving. In G. Gauthier, C. Frasson & K. VanLehn (Eds.), *Proceedings of the 5th International Conference on Intelligent Tutoring Systems* (pp. 33–42). Berlin, Heidelberg, Germany: Springer.

Gardner, J., Brooks, C., & Baker, R. (2019). Evaluating the fairness of predictive student models through slicing analysis. *Proceedings of the 9th International Conference on Learning Analytics & Knowledge* (pp. 225–234). New York: ACM.

Gasparetti, F., Limongelli, C., & Sciarrone, F. (2015). Exploiting Wikipedia for discovering prerequisite relationships among learning objects. *International Conference on Information Technology Based Higher Education and Training (ITHET)* (pp. 1–6).

Gauthier, G., Lajoie, S., Richard, S. & Wiseman, J. (2007). Mapping and validating diagnostic reasoning through interactive case creation. In T. Bastiaens & S. Carliner (Eds.), *Proceedings of E-Learn 2007—World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education* (pp. 2553–2562). Quebec City, Canada: Association for the Advancement of Computing in Education (AACE).

Geden, M., Emerson, A., Carpenter, D., Rowe, J., Azevedo, R., & Lester, J. (2021). Predictive student modeling in game-based learning environments with word embedding representations of reflection. *International Journal of Artificial Intelligence in Education*, *31*(1), 1–23.

Glassman, E. L., & Miller, R. C. (2016). Leveraging learners for teaching programming and hardware design at scale. *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion* (pp. 37–40).

Gobert, J. D., Pedro, M. S., Raziuddin, J., & Baker, R. S. (2013). From log files to assessment metrics: Measuring students' science inquiry skills using educational data mining. *Journal of the Learning Sciences*, *22*(4), 521–563.

Goguadze, G., Sosnovsky, S. A., Isotani, S., & McLaren, B. M. (2011). Evaluating a Bayesian student model of decimal misconceptions. *International Conference on Educational Data Mining* (pp. 301–306).

González-Brenes, J., Huang, Y., & Brusilovsky, P. (2014). General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. *Proceedings of the 7th International Conference on Educational Data Mining* (pp. 84–91).

Ha, E. Y., Rowe, J., Mott, B., & Lester, J. (2011). Goal recognition with Markov logic networks for player-adaptive games. *Proceedings of the 7th Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (pp. 32–39). Menlo Park, CA: AAAI.

Hagemans, M. G., van der Meij, H., & de Jong, T. (2013). The effects of a concept map-based support tool on simulation-based inquiry learning. *Journal of Educational Psychology*, *105*(1), 1–24.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10–18.

Heffernan, N. T., & Heffernan, C. L. (2014). The ASSISTments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, *24*(4), 470–497.

Heffernan, N. T., Koedinger, K. R., & Razzaq, L. (2008). Expanding the model-tracing architecture: A 3rd generation intelligent tutor for algebra symbolization. *International Journal of Artificial Intelligence in Education*, *18*(2), 153–178.

Heffernan, N. T., Ostrow, K. S., Kelly, K., Selent, D., van Inwegen, E. G., Xiong, X., & Williams, J. J. (2016). The future of adaptive learning: Does the crowd hold the key? *International Journal of Artificial Intelligence in Education*, *26*(2), 615–644.

Heller, J., Steiner, C., Hockemeyer, C., & Albert, D. (2006). Competence-based knowledge structures for personalised learning. *International Journal on E-learning*, *5*(1), 75–88.

Henze, N., & Nejdl, W. (1999). Adaptivity in the KBS hyperbook system. *The 2nd Workshop on Adaptive Systems and User Modeling on the WWW.*

Holmes, W., Bialik, M., & Fadel, C. (2019). *Artificial intelligence in education: Promises and implications for teaching and Learning*. Boston, MA: The Center for Curriculum Redesign.

Huang, Y. (2018). *Learner modeling for integration skills in programming* (Doctoral dissertation). University of Pittsburgh, Pittsburgh, PA.

Huang, Y., González-Brenes, J. P., & Brusilovsky, P. (2015). Challenges of using observational data to determine the importance of example usage. *International Conference on Artificial Intelligence in Education* (pp. 633–637). Cham, Switzerland: Springer.

Huang, Y., González-Brenes, J. P., Kumar, R., & Brusilovsky, P. (2015). A framework for multifaceted evaluation of student models. *Proceedings of the 8th International Conference on Educational Data Mining* (pp. 203–210).

Huang, Y., Guerra-Hollstein, J. P., Barria-Pineda, J., & Brusilovsky, P. (2017). Learner modeling for integration skills. *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization* (pp. 85–93).

Huang, Y., Lobczowski, N. G., Richey, J. E., McLaughlin, E. A., Asher, M. W., Harackiewicz, J., … Koedinger, K. R. (2021). A general multi-method approach to data-driven redesign of tutoring systems. *Proceedings of the 11th International Conference on Learning Analytics and Knowledge* (pp. 161–172).

Huang, Y., Yudelson, M., Han, S., He, D., & Brusilovsky, P. (2016). A framework for dynamic knowledge modeling in textbook-based learning. *Proceedings of the 2016 conference on User Modeling, Adaptation and Personalization* (pp. 141–150).

Jiang, Y., Bosch, N., Baker, R. S., Paquette, L., Ocumpaugh, J., Andres, J. M. A. L., ... Biswas, G. (2018). Expert feature-engineering vs. deep neural networks: Which is better for sensor-free affect detection? *Proceedings of the 19th International Conference on Artificial Intelligence in Education* (pp. 198–211). Springer, Cham.

Joyner, D. A., Majerich, D. M., & Goel, A. K. (2013). Facilitating authentic reasoning about complex systems in middle school science education. *Procedia Computer Science*, *16*, 1043–1052.

Käser, T., Klingler, S., Schwing, A. G., & Gross, M. (2014). Beyond knowledge tracing: Modeling skill topologies with Bayesian networks. *International Conference on Intelligent Tutoring Systems* (pp. 188–198).

Käser, T., & Schwartz, D. L. (2020). Modeling and analyzing inquiry strategies in open-ended learning environments. *International Journal of Artificial Intelligence in Education*, *30*(3), 504–535.

Kavcic, A. (2004). Fuzzy user modeling for adaptation in educational hypermedia. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *34*(4), 439–449.

Kelly, D. P., & Rutherford, T. (2017). Khan Academy as supplemental instruction: A controlled study of a computer-based mathematics intervention. *The International Review of Research in Open and Distributed Learning*, *18*(4). doi: 10.19173/irrodl.v18i4.2984

Kim, J. (2015). *Learnersourcing: Improving learning with collective learner activity* (Doctoral dissertation). Massachusetts Institute of Technology, Cambridge, MA.

Kinnebrew, J. S., Loretz, K. M., & Biswas, G. (2013). A contextualized, differential sequence mining method to derive students' learning behavior patterns. *Journal of Educational Data Mining*, *5*(1), 190–219.

Kluger, A. N., & DeNisi, A. (1996). The effects of feedback interventions on performance: A historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psychological Bulletin*, *119*(2), 254.

Koedinger, K. R., & Aleven, V. (2007). Exploring the assistance dilemma in experiments with Cognitive Tutors. *Educational Psychology Review*, *19*(3), 239–264.

Koedinger, K. R., & Aleven, V. (2021). Multimedia learning with cognitive tutors. To appear in R. E. Mayer & L. Fiorella (Eds.), *The Cambridge handbook of multimedia learning* (3rd ed). Cambridge, UK: Cambridge University Press.

Koedinger, K. R., Aleven, V., & Heffernan, N. (2003). Toward a rapid development environment for Cognitive Tutors. *Proceedings of the International Conference on Artificial Intelligence in Education* (pp. 455–457).

Koedinger, K. R., Baker, R. S., Cunningham, K., Skogsholm, A., Leber, B., & Stamper, J. (2010). A data repository for the EDM community: The PSLC DataShop. *Handbook of Educational Data Mining*, *43*, 43–56.

Koedinger, K. R., Brunskill, E., Baker, R. S., McLaughlin, E. A., & Stamper, J. (2013). New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, *34*(3), 27–41.

Koedinger, K. R., & Corbett, A. T. (2006). Cognitive Tutors: Technology bringing learning sciences to the classroom. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (pp. 61–78). New York: Cambridge University Press.

Koedinger, K. R., Corbett, A. T., & Perfetti, C. (2012). The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science*, *36*(5), 757–798.

Koedinger, K., Liu, R., Stamper, J., Thille, C., & Pavlik, P. (2017). Community based educational data repositories and analysis tools. *Proceedings of the Seventh International Conference on Learning Analytics & Knowledge* (pp. 524–525).

Koedinger, K., & McLaughlin, E. (2010). Seeing language learning inside the math: Cognitive analysis yields transfer. *Proceedings of the Annual Meeting of the Cognitive Science Society*, *32*(32), 471–476.

Koedinger, K. R., McLaughlin, E. A., & Stamper, J. C. (2012). Automated student model improvement. *Proceedings of the 5th International Conference on Educational Data Mining* (pp. 17–24).

Koedinger, K.R., Stamper, J.C., McLaughlin, E.A., & Nixon, T. (2013). Using data-driven discovery of better student models to improve student learning. In H.C. Lane, K. Yacef, J. Mostow & P. Pavlik (Eds.), *International Conference on Artificial Intelligence in Education* (pp. 421–430). Berlin, Heidelberg, Germany: Springer.

Kotu, V., & Deshpande, B. (2014). *Predictive analytics and data mining: Concepts and practice with rapidminer*. Morgan Kaufmann.

Labutov, I., Huang, Y., Brusilovsky, P., & He, D. (2017). Semi-supervised techniques for mining learning outcomes and prerequisites. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 907–915).

Lenat, D. B., & Durlach, P. J. (2014). Reinforcing math knowledge by immersing students in a simulated learning-by-teaching experience. *International Journal of Artificial Intelligence in Education*, *24*(3), 216–250.

Lindsey, R. V., Khajah, M., & Mozer, M. C. (2014). Automatic discovery of cognitive skills to improve the prediction of student learning. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence & K.

Q. Weinberger (Eds.), *Proceedings of the 28th International Conference on Advances in Neural Information Processing Systems* (pp. 1386–1394).

Li, H., Gobert, J., Dickler, R., & Moussavi, R. (2018). The impact of multiple real-time scaffolding experiences on science inquiry practices. *International Conference on Intelligent Tutoring Systems* (pp. 99–109). Springer, Cham.

Li, N., Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2015). Integrating representation learning and skill learning in a human-like intelligent agent. *Artificial Intelligence*, *219*, 67–91.

Liu, C., Kim, J., & Wang, H. C. (2018). Conceptscape: Collaborative concept mapping for video learning. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (pp. 1–12).

Liu, R., & Koedinger, K. R. (2017). Closing the loop: Automated data-driven cognitive model discoveries lead to improved instruction and learning gains. *Journal of Educational Data Mining*, *9*(1), 25–41.

Livieris, I. E., Drakopoulou, K., Tampakas, V. T., Mikropoulos, T. A., & Pintelas, P. (2019). Predicting secondary school students' performance utilizing a semi-supervised learning approach. *Journal of Educational Computing Research*, *57*(2), 448–470.

Lodder, J., Heeren, B., Jeuring, J., & Neijenhuis, W. (2021). Generation and use of hints and feedback in a Hilbert-style axiomatic proof tutor. *International Journal of Artificial Intelligence in Education*, *31*(1), 99–133.

Long, Y., & Aleven, V. (2014). Gamification of joint student/system control over problem selection in a linear equation tutor. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Proceedings of the 12th International Conference on Intelligent Tutoring Systems*, *ITS 2014* (pp. 378–387). New York: Springer.

Lovett, M. C. (1998). Cognitive task analysis in service of intelligent tutoring system design: A case study in statistics. In *International Conference on Intelligent Tutoring Systems* (pp. 234–243). Berlin, Heidelberg, Germany: Springer.

Lynch, C., Ashley, K. D., Pinkwart, N., & Aleven, V. (2009). Concepts, structures, and goals: Redefining ill-definedness. *International Journal of Artificial Intelligence in Education*, *19*(3), 253–266.

MacLellan, C. J., & Koedinger, K. R. (2020). Domain-general tutor authoring with Apprentice Learner models. *International Journal of Artificial Intelligence in Education*.

Martinez-Maldonado, R., Clayphan, A., Yacef, K., & Kay, J. (2014). MTFeedback: Providing notifications to enhance teacher awareness of small group work in the classroom. *IEEE Transactions on Learning Technologies*, *8*(2), 187–200.

Martínez-Maldonado, R., Kay, J., & Yacef, K. (2010, November). Collaborative concept mapping at the tabletop. In *ACM International Conference on Interactive Tabletops and Surfaces* (pp. 207–210).

Mathan, S. A., & Koedinger, K. R. (2005). Fostering the intelligent novice: Learning from errors with metacognitive tutoring. *Educational Psychologist*, *40*(4), 257–265.

Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2015). Teaching the teacher: Tutoring SimStudent leads to more effective cognitive tutor authoring. *International Journal of Artificial Intelligence in Education*, *25*(1), 1–34.

Matsuda, N., Furukawa, T., Bier, N., & Faloutsos, C. (2015). Machine beats experts: Automatic discovery of skill models for data-driven online course refinement. *Proceedings of the 8th International Conference on Educational Data Mining* (pp. 101–108).

Matsuda, N., Weng, W., & Wall, N. (2020). The effect of metacognitive scaffolding for learning by teaching a teachable agent. *International Journal of Artificial Intelligence in Education*, 1–37.

Matsuda, N., Yarzebinski, E., Keiser, V., Raizada, R., Cohen, W. W., Stylianides, G. J., & Koedinger, K. R. (2013). Cognitive anatomy of tutor learning: Lessons learned with SimStudent. *Journal of Educational Psychology*, *105*(4), 1152–1163.

Mayo, M., & Mitrovic, A., (2001) Optimising ITS behaviour with Bayesian networks and decision theory. *International Journal on Artificial Intelligence in Education*, *12*(2), 124–153.

McKendree, J. (1990). Effective feedback content for tutoring complex skills. *Human-Computer Interaction*, *5*(4), 381–413.

McLaren, B. M., van Gog, T., Ganoe, C., Karabinos, M., & Yaron, D. (2016). The efficiency of worked examples compared to erroneous examples, tutored problem solving, and problem solving in computer-based learning environments. *Computers in Human Behavior*, *55*, 87–99.

Means, B., & Gott, S. P. (1988). Cognitive task analysis as a basis for tutor development: Articulating abstract knowledge representations. In J. Psotka, L. D. Massey & S. A. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned* (pp. 35–57). Lawrence Erlbaum Associates.

Melis, E., Andres, E., Büdenbender, J., Frischauf, A., Goguadze, G., Libbrecht, P., … & Ullrich, C. (2001). ActiveMath: A generic and adaptive web-based learning environment. *International Journal of Artificial Intelligence in Education*, *12*(4), 385–407.

Michalenko, J. J., Lan, A. S., & Baraniuk, R. G. (2017). Data-mining textual responses to uncover misconception patterns. *Proceedings of the Fourth ACM Conference on Learning @ Scale* (pp. 245–248).

Millán, E., Loboda, T., & Pérez-De-La-Cruz, J. L. (2010). Bayesian networks for student model engineering. *Computers & Education*, *55*(4), 1663–1683.

Millán, E., & Pérez-de-la Cruz, J.-L. (2002). A Bayesian diagnostic algorithm for student modeling and its evaluation. *User Modeling and User-Adapted Interaction*, *12*(2), 281–330.

Min, W., Frankosky, M., Mott, B., Rowe, J., Smith, A., Wiebe, E., … Lester, J. (2020). DeepStealth: Game-based learning stealth assessment with deep neural networks. *IEEE Transactions on Learning Technologies*, *13*(2), 312–325.

Min, W., Mott, B., Rowe, J., Liu, B., & Lester, J. (2016). Player goal recognition in open-world digital games with long short-term memory networks. *Proceedings of the 25th International Joint Conference on Artificial Intelligence* (pp. 2590–2596).

Mislevy, R. J., & Gitomer, D. H. (1995). The role of probability-based inference in an intelligent tutoring system. *ETS Research Report Series*, *1995*(2), i-27.

Mitrovic, A. (1998). Experiences in implementing constraint-based modeling in SQL-Tutor. In B. Goettl, H. Halff, C. Redfield & V. Shute (Eds.), *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 414–423). Springer.

Mitrovic, A. (2003). An intelligent SQL tutor on the Web. *International Journal of Artificial Intelligence in Education*, *13*(2–4), 173–197.

Mitrovic, A. (2010). Modeling domains and students with constraint-based modeling. In R. Nkambou, J. Bordeaux & R. Mizoguchi (Eds.), *Advances in Intelligent Tutoring Systems. Studies in Computational Intelligence* (pp. 63–80). Berlin, Heidelberg, Germany: Springer.

Mitrovic, A. (2012). Fifteen years of constraint-based tutors: What we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, *22*(1–2), 39–72.

Mitrovic, A., Koedinger, K., & Martin, B. (2003). A comparative analysis of cognitive tutoring and constraint-based modelling. In P. Brusilovsky, A. Corbett & F. de Rosis (Eds.), *Proceedings of the 9th International Conference on User Modeling* (pp. 313–322). Berlin, Heidelberg, Germany: Springer.

Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., & McGuigan, N. (2009). ASPIRE: An authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, *19*(2), 155–188.

Mitrovic, A., Mathews, M., Ohlsson, S., Holland, J., & McKinlay, A. (2016). Computer-based post-stroke rehabilitation of prospective memory. *Journal of Applied Research in Memory and Cognition*, *5*(2), 204–214.

Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence in Education*, *10*(3–4), 238–256.

Mitrovic, A., & Ohlsson, S. (2006). Constraint-based knowledge representation for individualized instruction. *Computer Science and Information Systems*, *3*(1), 1–22.

Mitrovic, A., Ohlsson, S., & Barrow, D. (2013). The effect of positive feedback in a constraint-based intelligent tutoring system. *Computers & Education*, *60*(1), 264–272.

Mitrovic, A., & Weerasinghe, A. (2009). Revisiting the ill-definedness and consequences for ITSs. In V. Dimitrova, R. Mizoguchi, B. du Boulay, A. Graesser (Eds.), *Proceedings of the 14th International Conference on Artificial Intelligence in Education* (pp. 375–382).

Moore, S., Nguyen, H. A., & Stamper, J. (2020). Towards crowdsourcing the identification of knowledge components. *Proceedings of the Seventh ACM Conference on Learning@ Scale* (pp. 245–248).

Mott, B., Lee, S., & Lester, J. (2006). Probabilistic goal recognition in interactive narrative environments. In Gil, Y. & Mooney, R. (Eds.), *Proceedings of the Twenty-First National Conference on Artificial Intelligence* (pp. 187–192).

Murray, R. C., VanLehn, K., & Mostow, J. (2004). Looking ahead to select tutorial actions: A decision-theoretic approach. *International Journal of Artificial Intelligence in Education*, *14*(3, 4), 235–278.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Novak, J. D. (1996). Concept mapping: A tool for improving science teaching and learning. *Improving Teaching and Learning in Science and Mathematics*, 32–43.

Ohlsson, S. (1996). Learning from performance errors. *Psychological Review*, *103*(2), 241.

Ohlsson, S., & Mitrovic, A. (2007). Fidelity and efficiency of knowledge representations for intelligent tutoring systems. *Technology, Instruction, Cognition and Learning*, *5*(2), 101–132.

Olsen, J. K., Rummel, N., & Aleven, V. (2019). It is not either or: An initial investigation into combining collaborative and individual learning using an ITS. *International Journal of Computer-Supported Collaborative Learning*, *14*(3), 353–381.

Pane, J. F., Griffin, B. A., McCaffrey, D. F., & Karam, R. (2014). Effectiveness of Cognitive Tutor Algebra I at scale. *Educational Evaluation and Policy Analysis*, *36*(2), 127–144.

Pane, J. F., McCaffrey, D. F., Slaughter, M. E., Steele, J. L., & Ikemoto, G. S. (2010). An experiment to evaluate the efficacy of cognitive tutor geometry. *Journal of Research on Educational Effectiveness*, *3*(3), 254–281.

Paquette, L., & Baker, R. (2019). Comparing machine learning to knowledge engineering for student behavior modeling: A case study in gaming the system. *Interactive Learning Environments*, *27*(5–6), 585–597.

Pardos, Z. A., & Dadu, A. (2017). Imputing KCs with representations of problem content and context. *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization* (pp. 148–155).

Pardos, Z. A., Heffernan, N. T., Anderson, B., & Heffernan, C. L. (2006). Using fine-grained skill models to fit student performance with Bayesian networks. *Proceedings of the Workshop in Educational Data Mining held at the 8th International Conference on Intelligent Tutoring Systems*. Taiwan, China.

Park, D. H., Hendricks, L. A., Akata, Z., Rohrbach, A., Schiele, B., Darrell, T., & Rohrbach, M. (2018). Multimodal explanations: Justifying decisions and pointing to the evidence. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8779–8788).

Pavlik, P.I ., Brawner, K., Olney, A., & Mitrovic, A. (2013). A review of student models used in intelligent tutoring systems. In R. A. Sottilare, A. Graesser, X. Hu & H. Holden (Eds), *Design Recommendations for Intelligent Tutoring Systems*, *1*, 39–68. Orlando, FL: US Army Research Laboratory.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

Pelánek, R. (2017). Bayesian knowledge tracing, logistic models, and beyond: An overview of learner modeling techniques. *User Modeling and User-Adapted Interaction*, *27*(3–5), 313–350.

Pinkwart, N., Ashley, K., Lynch, C., & Aleven, V. (2009). Evaluating an intelligent tutoring system for making legal arguments with hypotheticals. *International Journal of Artificial Intelligence in Education*, *19*(4), 401–424.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Rafferty, A. N., & Griffiths, T. L. (2015). Interpreting freeform equation solving. *Proceedings of the 17th International Conference on Artificial Intelligence in Education* (pp. 387–397). Springer, Cham.

Rafferty, A. N., Jansen, R., & Griffiths, T. L. (2016). Using inverse planning for personalized feedback. In T. Barnes, M. Chi & M. Feng (Eds.), *Proceedings of the 9th International Conference on Educational Data Mining* (pp. 472–477).

Reilly, J. M., & Dede, C. (2019). Differences in student trajectories via filtered time series analysis in an immersive virtual world. *Proceedings of the 9th International Conference on Learning Analytics & Knowledge* (pp. 130–134).

Reye, J. (1996). A belief net backbone for student modelling. *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 596–604). Berlin, Heidelberg, Germany: Springer.

Ritter, S. (1997). Communication, cooperation and competition among multiple tutor agents. In B. du Boulay & R. Mizoguchi (Eds.), *Artificial Intelligence in Education*, *Proceedings of AI-ED 97 World Conference* (pp. 31–38). Amsterdam, Netherlands: IOS Press.

Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. (2007). Cognitive Tutor: Applied research in mathematics education. *Psychonomic Bulletin & Review*, *14*(2), 249–255.

Rivers, K., & Koedinger, K. R. (2017). Data-driven hint generation in vast solution spaces: A self-improving Python programming tutor. *International Journal of Artificial Intelligence in Education*, *27*(1), 37–64.

Roll, I., Aleven, V., & Koedinger, K. R. (2010). The invention lab: Using a hybrid of model tracing and constraint-based modeling to offer intelligent support in inquiry environments. In *International Conference on Intelligent Tutoring Systems* (pp. 115–124). Berlin, Heidelberg, Germany: Springer.

Rosé, C. P., McLaughlin, E. A., Liu, R., & Koedinger, K. R. (2019). Explanatory learner models: Why machine learning (alone) is not the answer. *British Journal of Educational Technology*, *50*(6), 2943–2958.

Rowe, J., & Lester, J. (2010). Modeling user knowledge with dynamic Bayesian networks in interactive narrative environments. *Proceedings of the Sixth Annual Artificial Intelligence and Interactive Digital Entertainment* (pp. 57–62).

Rowe, J. & Lester, J. (2015). Improving student problem solving in narrative-centered learning environments: A modular reinforcement learning framework. Proceedings *of the Seventeenth International Conference on Artificial Intelligence in Education* (pp. 419–428).

Russell, S., & Norvig, P. (2020). *Artificial intelligence: A modern approach* (Fourth Edition). Pearson.

Salden, R. J., Aleven, V., Schwonke, R., & Renkl, A. (2010). The expertise reversal effect and worked examples in tutored problem solving. *Instructional Science*, *38*(3), 289–307.

Salden, R. J., Koedinger, K. R., Renkl, A., Aleven, V., & McLaren, B. M. (2010). Accounting for beneficial effects of worked examples in tutored problem solving. *Educational Psychology Review*, *22*(379–392). doi: 10.1007/s10648-010-9143-6.

Sao Pedro, M. A., Baker, R., Gobert, J. D., Montalvo, O., & Nakama, A. (2013). Leveraging machine-learned detectors of systematic inquiry behavior to estimate and predict transfer of inquiry skill. *User Modeling and User-Adapted Interaction*, *23*(1), 1–39.

Sawyer, R., Rowe, J., Azevedo, R., & Lester, J. (2018). Filtered time series analyses of student problem-solving behaviors in game-based learning. *Proceedings of the Eleventh International Conference on Educational Data Mining* (pp. 229–238).

Schank, R. C. (1990). Case-based teaching: Four experiences in educational software design. *Interactive Learning Environments*, *1*(4), 231–253.

Schwartz, D. L., Chase, C. C., Oppezzo, M. A., & Chin, D. B. (2011). Practicing versus inventing with contrasting cases: The effects of telling first on learning and transfer. *Journal of Educational Psychology*, *103*(4), 759.

Shen, S., Mostafavi, B., Barnes, T., & Chi, M. (2018). Exploring induced pedagogical strategies through a Markov decision process framework: Lessons learned. *Journal of Educational Data Mining*, *10*(3), 27–68.

Shi, Y., Shah, K., Wang, W., Marwan, S., Penmetsa, P., & Price, T. (2021). Toward semi-automatic misconception discovery using code embeddings. *Proceedings of the 11th International Conference on Learning Analytics and Knowledge* (pp. 606–612).

Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research*, *78*(1), 153–189.

Shute, V. J., Wang, L., Greiff, S., Zhao, W., & Moore, G. (2016). Measuring problem solving skills via stealth assessment in an engaging video game. *Computers in Human Behavior*, 63, 106–117.

Sklavakis, D., & Refanidis, I. (2013). Mathesis: An intelligent web-based algebra tutoring school. *International Journal of Artificial Intelligence in Education*, *22*(4), 191–218.

Slater, S., Baker, R., Almeda, M. V., Bowers, A., & Heffernan, N. (2017). Using correlational topic modeling for automated topic identification in intelligent tutoring systems. *Proceedings of the Seventh International Learning Analytics & Knowledge Conference* (pp. 393–397).

Sleeman, D., Kelly, A. E., Martinak, R., Ward, R. D., & Moore, J. L. (1989). Studies of diagnosis and remediation with high school algebra students. *Cognitive Science*, *13*(4), 551–568.

Sosnovsky, S., & Dicheva, D. (2010). Ontological technologies for user modelling. *International Journal of Metadata, Semantics and Ontologies*, *5*(1), 32–71.

Sosnovsky, S., Brusilovsky, P., Yudelson, M., Mitrovic, A., Mathews. M., & Kumar, A. (2009). Semantic integration of adaptive educational systems. In T. Kuflik, S. Berkovsky, F. Carmagnola & D. Heckmann (Eds.), *Advances in Ubiquitous User Modelling: Revised Selected Papers, Springer LCNS*, 5830, 134–158.

Sosnovsky, S., Hsiao, I. H., & Brusilovsky, P. (2012). Adaptation "in the wild": Ontology-based personalization of open-corpus learning material. In *European Conference on Technology Enhanced Learning* (pp. 425–431). Berlin, Heidelberg, Germany: Springer.

Sottilare, R. A., Graesser, A. C., Hu, X., Olney, A., Nye, B., & Sinatra, A. M. (Eds.). (2016). *Design recommendations for Intelligent Tutoring Systems: Volume 4-Domain Modeling* (Vol. 4). Orlando, FL: US Army Research Laboratory.

Sowa, J. (1994). *Conceptual structures: Information processing in mind and machine*. Boston, MA: Addison-Wesley.

Stacey, K., Sonenberg, E., Nicholson, A., Boneh, T., & Steinle, V. (2003). A teaching model exploiting cognitive conflict driven by a Bayesian network. *International Conference on User Modeling* (pp. 352–362). Berlin, Heidelberg, Germany: Springer.

Stamper, J., Eagle, M., Barnes, T., & Croy, M. (2013). Experimental evaluation of automatic hint generation for a logic tutor. *International Journal of Artificial Intelligence in Education*, *22*(1–2), 3–17.

Sukthankar, G., Geib, C., Bui, H. H., Pynadath, D., & Goldman, R. P. (Eds.). (2014). *Plan, activity, and intent recognition: Theory and practice*. San Francisco, CA: Morgan Kaufman.

Suraweera, P., Mitrovic, A., & Martin, B. (2010). Widening the knowledge acquisition bottleneck for constraint-based tutors. *International Journal on Artificial Intelligence in Education*, *20*(2), 137–173.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Tatsuoka, K. K. (1983). Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, *20*(4), 345–354.

Taub, M., Azevedo, R., Bradbury, A., Millar, G., & Lester, J. (2018). Using sequence mining to reveal the efficiency in scientific reasoning during STEM learning with a game-based learning environment. *Learning and Instruction*, *54*, 93–103.

Thaker, K., Huang, Y., Brusilovsky, P., & Daqing, H. (2018). Dynamic knowledge modeling with heterogeneous activities for adaptive textbooks. *The 11th International Conference on Educational Data Mining* (pp. 592–595).

Tofel-Grehl, C., & Feldon, D. F. (2013). Cognitive task analysis–based training: A meta-analysis of studies. *Journal of Cognitive Engineering and Decision Making*, *7*(3), 293–304.

van der Kleij, F. M., Feskens, R. C., & Eggen, T. J. (2015). Effects of feedback in a computer-based learning environment on students' learning outcomes: A meta-analysis. *Review of Educational Research*, *85*(4), 475–511.

Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109*(2), 373–440.

VanLehn, K. (1990). *Mind bugs: The origins of procedural misconceptions*. Cambridge, MA: MIT Press.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, *46*(4), 197–221.

VanLehn, K., Burkhardt, H., Cheema, S., Kang, S., Pead, D., Schoenfeld, A., & Wetzel, J. (2021). Can an orchestration system increase collaborative, productive struggle in teaching-by-eliciting classrooms? *Interactive Learning Environments*, 29(6), 987–1005.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., … Wintersgill, M. (2005). The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education*, *15*(3), 147–204.

Vomlel, J. (2004). Bayesian networks in educational testing. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, *12*, 83–100.

Von Ahn, L. (2013). Duolingo: learn a language for free while helping to translate the web. *Proceedings of the 2013 International Conference on Intelligent User Interfaces* (pp. 1–2).

Walker, E., Rummel, N., & Koedinger, K. R. (2014). Adaptive intelligent support to improve peer tutoring in algebra. *International Journal of Artificial Intelligence in Education*, *24*(1), 33–61.

Wang, S., Liang, C., Wu, Z., Williams, K., Pursel, B., Brautigam, B., … Giles, C. L. (2015). Concept hierarchy extraction from textbooks. *Proceedings of the 2015 ACM Symposium on Document Engineering* (pp. 147–156).

Weber, G., & Brusilovsky, P. (2001). ELM-ART: An adaptive versatile system for web-based instruction. *International Journal of Artificial Intelligence in Education*, *12*, 351–384.

Weerasinghe, A., & Mitrovic, A. (2006). Facilitating deep learning through self-explanation in an open-ended domain. *International Journal of Knowledge-based and Intelligent Engineering Systems*, *10*(1), 3–19.

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

Weir, S., Kim, J., Gajos, K. Z., & Miller, R. C. (2015). Learnersourcing subgoal labels for how-to videos. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (pp. 405–416).

Weitekamp, D., Harpstead, E., & Koedinger, K. R. (2020). An interaction design for machine teaching to develop AI tutors. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–11).

Williams, J. J., Kim, J., Rafferty, A., Maldonado, S., Gajos, K. Z., Lasecki, W. S., & Heffernan, N. (2016). Axis: Generating explanations at scale with learnersourcing and machine learning. *Proceedings of the Third (2016) ACM Conference on Learning@ Scale* (pp. 379–388).

Yang, K. B., Nagashima, T., Yao, J., Williams, J. J., Holstein, K., & Aleven, V. (2021). Can crowds customize instructional materials with minimal expert guidance? Exploring teacher-guided crowdsourcing for improving hints in an AI-based tutor. *Proceedings of the ACM on Human-Computer Interaction*, *5*(CSCW1), 1–24.

Xu, Y., & Mostow, J. (2012). Comparison of methods to trace multiple subskills: Is LR-DBN best? *Proceedings of the 5th International Conference on Educational Data Mining* (pp. 41–48).

Zhi, R., Price, T. W., Marwan, S., Milliken, A., Barnes, T., & Chi, M. (2019). Exploring the impact of worked examples in a novice programming environment. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 98–104).