AnalyticalInk: An Interactive Learning Environment for Math Word Problem Solving

Bo Kang
University of Central Florida
Orlando, FL 32816 USA
bkang@cs.ucf.edu

Arun Kulshreshth
University of Central Florida
Orlando, FL 32816 USA
arunkul@knights.ucf.edu

Joseph J. LaViola Jr
University of Central Florida
Orlando, FL 32816 USA
jjl@eecs.ucf.edu

ABSTRACT
We present *AnalyticalInk*, a novel math learning environment prototype that uses a semantic graph as the knowledge representation of algebraic and geometric word problems. The system solves math problems by reasoning upon the semantic graph and automatically generates conceptual and procedural scaffoldings in sequence. We further introduces a step-wise tutoring framework, which can check students' input steps and provide the adaptive scaffolding feedback. Based on the knowledge representation, *AnalyticalInk* highlights keywords that allow users to further drag them onto the workspace to gather insight into the problem's initial conditions. The system simulates a pen-and-paper environment to let users input both in algebraic and geometric workspaces. We conducted an usability evaluation to measure the effectiveness of *AnalyticalInk*. We found that keyword highlighting and dragging is useful and effective toward math problem solving. Answer checking in the tutoring component is useful. In general, our prototype shows the promise in helping users to understand geometrical concepts and master algebraic procedures under the problem solving.

Author Keywords
Semantic Representation; Inference and Reasoning; Scaffolding Generation; User Modeling; Intelligent Tutoring System; Learning Techniques.

ACM Classification Keywords
H.5.2. User Interfaces: Graphical user interfaces (GUI).; I.2.4. Knowledge Representation Formalisms and Methods: Representations.; I.2.1. Applications and Expert Systems: Natural language interfaces.

INTRODUCTION
Mathematical word problem solving is a complex task which requires users to abstract the symbolic knowledge representation to formulate the math concept, manipulate and transform symbolic knowledge, and infer unknown knowledge [30]. As the quantitative representation of such problems changes, the solving task becomes more difficult, which affects users' reasoning process [16, 21].

IUI'16, March 07-10, 2016, Sonoma, CA, USA.

DOI: http://dx.doi.org/10.1145/2856767.2856789

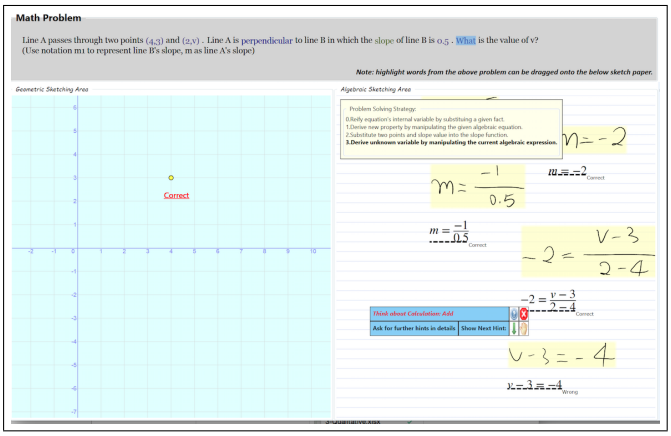


Figure 1: An example scenario where a user wants to solve the given math word problem on the dual geometric-algebraic workspace. In *AnalyticalInk*, the user drags the highlight words from the problem statement onto the below workspace to gain insight of problem's initial condition. Further, the user can sketch solving steps and query the system to check their correctness. As the current step is recognized a wrong, the system provides scaffoldings in visual widgets to guide the user's problem solving.

In this paper, we present *AnalyticalInk*, a novel math interactive learning environment prototype, which helps users to understand geometric concepts and solve geometry-and-algebra math problems. We illustrate a semantic graph knowledge representation to capture the key information of natural language based math word problems. The system relies on this representation to automatically solve the math problem and generates a scaffolding trace, which derives from known facts toward the unknown problem question as a sequence of scaffolding steps. We further demonstrates a step-wise tutoring framework upon the auto-generated scaffolding trace of a math problem. Such tutoring framework can check users' input steps and further provide adaptive scaffolding to help guiding a user through solving the problem.

The user interface of *AnalyticalInk* simulates the semantic word highlighting and dragging learning technique [8]. It lets users have a sense of the key information of a given problem and further allows them to drag information onto workspaces to either acquire the initial condition of a problem or reinforce geometric concept understanding. Interactive words directly map to annotated semantic words, which are defined by using the problem representation semantic. In addition, *AnalyticalInk* applies the theory of iterative development between the conceptual understanding and the procedural skill pruning from the math cognitive research [28].

The user interface provides both an algebraic canvas and a geometric canvas to let users determine the connection between geometric concepts and related algebraic procedures when solving geometry-and-algebra related problems. Lastly, *AnalyticalInk* supports sketch based user interactions, which have been shown to improve conceptual understanding, problem understanding and inferential reasoning during problem-solving [10, 25]. Figure 1 shows *AnalyticalInk*'s user interface. There are four contributions in this paper:

- A novel semantic graph based representation of geometry-and-algebra math problems is presented. We propose an automated reasoning process upon this knowledge representation to solve math word problems and generate scaffolding traces.
- A step-wise tutoring framework, which verifies users' input steps and further provides step-based adaptive scaffolding to guide user solving a problem.
- The novel learning environment *AnalyticalInk*, which embeds learning techniques to promote the math learning. The user interface of *AnalyticalInk* uses the sketch based natural input to let users reason during math problem-solving.
- The usability evaluation of *AnalyticalInk* is shown. The tutoring framework is evaluated and discussed. A comparative evaluation of the keyword highlighting and dragging learning technique is described.

RELATED WORK

Our work is closely related to three research areas: Computational Semantics of Natural Language, Intelligent Tutoring System (ITS), Learning based Interactive User Interface.

Computational Semantics and Reasoning

Our work considers the formal language representation of natural language descriptions. Specifically, both frame semantics and abstract meaning representation (AMR) are related to this work and have been explored in the natural language processing community. Frame semantics is a linguistic theory that interprets the meaning of words based on their semantic frames [9]. Abstract meaning representation converts a sentence to a directed, acyclic graph (DAG) containing a set of nodes and edges [6]. The nodes correspond to concepts and edges as semantic relations. Our intermediate knowledge presentation of geometry-and-algebra math problem is directly inspired by the AMR problem semantic.

Previous works have been developing statistical methods to parse math word problems as template based equations and further employing a solver to obtain the solution [23, 31, 35]. Instead of template-based methods, our work presents an intermediate semantic to capture relations of math concepts. However, this work does not consider the semantic parsing or the word alignment problem. Instead, we directly annotate given geometry-and-algebra math problems with our defined problem semantic. Using the annotated problem semantic, we focus on simulating how students plan to solve a problem as a constraint searching process upon the semantic graph. Due to the human learning purpose, we also consider the sequential scaffoldings generation instead of retrieving a single problem answer during the above search process. Gulwani et.al. used the program synthesis to automatically solving the ruler/compass based geometry construction problem [13]. Our automatic solving and scaffolding generation is based on the search upon the semantic graph, which is similar to the program synthesis methodology. However, as geometric math problems are different, our goal is to generate scaffolding traces, which covers both geometrical concepts and algebraic procedures.

Intelligent Tutoring System

Intelligent Tutoring System (ITS) research attempts to design and integrate instructional scaffolding methodologies into the system to promote a deeper level of learning [19]. The core of ITS is a cognitive tutor, which applies production rules to build up the domain dependent expert system [3]. Model-tracing tutor is built upon the cognitive tutor, which provides the flag feedback, the buggy feedback and a chain of scaffoldings [15, 26]. The state-of-the-art algebra cognitive tutor includes procedural scaffolding to derive algebraic expression or equations [18]. In *AnalyticalInk*, besides procedural scaffoldings that derive math expressions and equations, the auto-generated scaffolding trace also includes conceptual scaffoldings to guide geometric concept matching, geometric relation detection and concept reification processes. *AnalyticalInk* can be categorized as a step-based tutoring system [32], which requires users to enter multiple steps to accomplish the problem solving process.

In addition, student modeling is another component of ITS to evaluate students' knowledge master level. Knowledge tracing is the state-of-the-art probabilistic graphical model to predict learning mastery per skill [7]. In this work, we do not consider how well users comprehend the certain knowledge component. Instead, we only check the correctness of user input steps and give the adaptive scaffolding feedback based on the matching result.

Learning based Math User Interface

Educational psychology research showed the evidence that highlighting keyword is one effective learning technique toward the natural language understanding [8]. Hefferman and Koedinger considered the problem acquisition from natural language narratives toward math expressions as a symbolization process [14]. They built a cognitive model in order to provide scaffoldings to guide this process. Instead of building a cognitive model, *AnalyticalInk* implements the keyword highlighting learning technique and further provides the dragging operation upon colored semantic keywords. Mathematical cognitive science research revealed that developing the conceptual understanding and the procedural skill follows an iterative process. They proved it by conducting the experiment using the decimal fraction domain knowledge. We applied this finding to guide the design of *AnalyticalInk*, which emphasizes the iterative view of geometric concepts and algebraic procedural skills development.

Sketch is a powerful means of interpersonal communication. Forbus et.al. illustrated the importance of sketching to support the conceptual knowledge, visual understanding, and language [10]. In addition, Oviatt et.al. described the term *inferential reasoning* from the interaction perspective to design

effective computational cognitive tools [25]. *AnalyticalInk* follows the sketch-based cognitive models to simulate pen-and-paper environment upon the user interface of interactive learning system. Previously, several pen-based math prototypes systems have been described, such as *MathPad*^{2} [24] ,*Hands-On Math* [33] and geometry theorem proving [17], Anthony et.al. demonstrated the effectiveness by integrating pen-based natural input with an intelligent tutoring system for algebra equation solving [5].

PROBLEM REPRESENTATION

When learning geometry concepts, students often encounter following types of math problems:

Problem1: Find the midpoint of the line segment joining $A(-2, 2)$ and $B(4, 6)$?

Problem2: There exists two points $A(2, 4)$ and $B(5, v)$, the distance between A and B is 5. What is the value of v?

Problem3: There are two points $A(2, y)$ and $B(-1, 4)$. The y-coordinate of point A is -1. What is the distance between these two points?

Problem4: Given an equation $2y+2x-y+2x+4 = 0$, graph this equation's corresponding shape? If it is a line, what is the slope of this line?

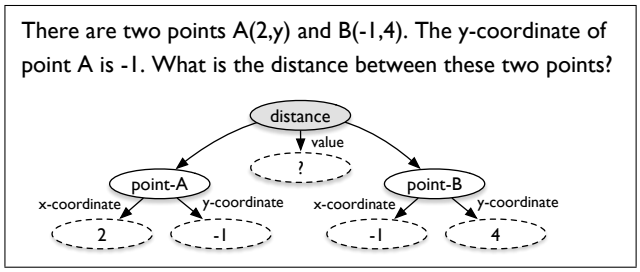
Problem5: A line passes through points $(2, 3)$ and $(4, v)$, the slope of this line is 5. What is the value of v? What is the y-intercept of the line?

Problem6: Line A passes through $(-1, 2)$ and $(5, 8)$. Line B is parallel to line A, and also crosses point $(1, 0)$, what is the general form of line B?

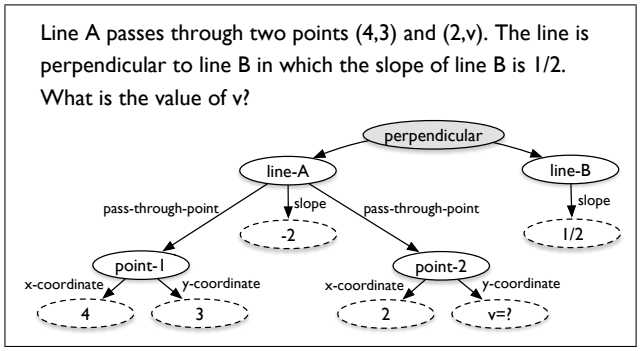
Problem7: Line A passes through two points $(4, 3)$ and $(2, v)$. The line is perpendicular to line B in which the slope of line B is $1/2$. what is the value of v?

These natural language narrative math problems all cover geometric concepts. When reasoning such concepts, algebraic procedural derivations are required to infer the same concept, or from one toward the other. The geometric concept understanding and the algebraic procedural derivation are performed in an iterative manner so as to solve such problems [28]. The geometric concept can be subdivided into three types of knowledge: *knowledge entity* (such as a point or a line), *knowledge attribute* (such as the y-coordinate of a point, or the slope of a line) and *knowledge relation* (such as two lines are parallel, one point is on the line).

The pattern of question and answer, in the above math problems, is deriving from several given concept instances toward unknown concept instances. This attributes to the planning of math problem solving [30]. For instance, Problem1 introduces two points and a line, and lets users derive the midpoint entity through the internal relation between these points. Problem6 shows two lines and three points; the point-and-line relation and the line-and-line relation are implicitly specified in the problem narrative. Users need to derive one line's general form from the given entities. In addition, Problem1, 3, 4 and 6 require users to build and reason knowledge relations both in the forward manner. However, Problem2, 5 and 7 let users to build relations in the forward manner, but reason for unknown geometric concepts in the backward manner.



(a) Problem 3's semantic DAG representation.



(b) Problem 7's semantic DAG representation.

Figure 2: Two math problems' semantic graph representations.

Syntax: We formulate the syntax of geometric concept knowledge as follows. All concept knowledge can be illustrated as the combination of a label, following by a colon and knowledge instance in the end. The label and colon is optional which depends on the problem description. For the *knowledge entity*, we currently model three entities: the point, the line and the circle. A point concept is represented in 2-dimensional Cartesian coordinates. The *knowledge attribute* is represented as the equation where the left side of it is a literal and the right side of the equation is a numeric value or an expression. The *knowledge relation* is represented as a function where the name of it is the relation and its arguments are knowledge entities. For the problem question, its syntax is an equation where the left side of it is a label or identity and the right side is a question mark. For instance, Problem3 contains two point knowledge entities as $A(2, y), B(-1, 4)$, a knowledge attribute $y = -1$, its question can be represented as $d =?$. Similarly, Problem7 contains two point entities $(4, 3), (2, v)$, two line entities $A : line, B : line$, a knowledge attribute $B : m = 1/2$, three knowledge relations $pass-through(A, (4, 3)), pass-through(A, (2, v)), perpendicular(A, B)$, its question can be represented as $v =?$.

Semantic Representation of Math Problems

In the semantic level, each math problem can be represented as a directed acyclic graph (DAG) using the syntax of geometric concepts. Figure 2 demonstrates two math word problems' corresponding DAG semantic representations. The graph contains a set of nodes and edges. The nodes are divided into three categories, including concepts, attributes,

and semantic relations. A concept (represented using an ellipse) corresponds to a semantic entity, such as a point or line. Multiple attributes (dashed ellipse) can be associated with a concept. For example, a point may have two attributes: x-coordinate and y-coordinate. A line may have multiple attributes including slope, y-intercept, pass-through-point, and corresponding equation. A semantic relation (shaded ellipse) captures relationship between two concepts. For example, Problem7 shows that line B is perpendicular to line A ('perpendicular'); Problem3 displays distance between point A and B ('distance'). A semantic relation may be associated with one or multiple attributes. A query term corresponds to an attribute with unknown value.

SYSTEM OVERVIEW

We illustrate the process to generate the semantic DAG per math problem, and describe the scaffolding trace synthesis during the semantic DAG generation. Based on the generated scaffolding trace, we present our model-tracing tutor. We show a hybrid match-and-verify framework upon the tutor, which can solve multiple solution paths' verification and further provide the adaptive scaffolding. Last, we show the sketch-based user interface of *AnalyticalInk*, which incorporates learning techniques: semantic word highlight/dragging and dual geometry-and-algebra problem solving.

Automatic Reasoning and Scaffolding Generation

Instead of semantic parsing geometry-and-algebra math problems, we annotate math word problems according to the syntax of geometric concepts. Each natural language math problem is expressed as a sequence of the annotated geometric concept. This annotation process simulates how students prime initial conditions of a math problem. Based on annotated syntax geometric concepts, we present our problem-solving plan approach to derive from known concepts toward unknown concepts. Figure 3 shows the higher level description to use the semantic DAG to simulate the problem solving plan process.

The scaffolding generation is executed in all four steps of problem-solving plan method. In detail, a scaffolding step or a trace step is synthesized by performing a pre-defined rule to derive a geometric concept from one state to another. After executing a sequence of rules, a scaffolding trace will be generated. In the scaffolding trace, there are two different types of generated scaffolding steps: the geometric concept scaffolding and the algebraic procedural scaffolding. We give further explanations of these two types and their relations in the following subsections.

Syntax Pattern Match

The goal of step 1 is to parse annotated geometric concepts as knowledge instances. Based on the geometry concept syntax, we use the parsing expression grammar (PEG) to parse geometric entities, attributes and relations [11]. In the syntax level, variations of geometric concept format exist. For example, the syntax level of a line concept contains the *line general form*, the *line point-slope form* or the *line slope-intercept form*. In addition, algebraic manipulations should be performed to transform an algebraic expression as a geometric concept. For example, the annotation

Input*:* Let **S** be a sequence of annotated syntax concepts.

Output*:* Let **G** be the semantic DAG representation.

Initiate G.

For every input s in S:

Step 1: geometric concept m = **Syntax Pattern Match**(s)

where m can be geometric entities, attributes or relations.

Step 2: semantic relations R = **Constraint-Checking** (G, m)

For every relation r in R:

Step 3: **Update-Graph**(G, r)

Step 4: **Graph Bottom-up Reification**(G)

Figure 3: A semantic DAG based constraint search method to simulate the human problem-solving plan process.

$2y + 2x - y + 2x + 4 = 0$ in Problem4 needs to be simplified using algebraic rule-derivations toward a recognizable line concept format $4x + y + 4 = 0$. To uniform the syntax of geometric concept, we build an algebraic rule-based expert system behind the PEG. Such expert system incorporates production rules with the arithmetic expression calculation and the algebraic expression manipulation.

In step 1, if the annotation matches the grammar without any rule-derivation, no scaffolding will be generated. However, if rules have been triggered to uniform the annotation as the geometric concept syntax format, both the geometric concept scaffolding and a sequence of algebraic procedural scaffoldings will be synthesized. The geometric scaffolding guides users to manipulate the algebraic expression so as to transform syntax annotations. Algebraic procedural scaffoldings are production rule derivations to explain this geometric scaffolding. Each procedural scaffolding or trace step records both geometric concept's formats before and after the rule derivation with the production rule itself. For example, transforming $2y + 2x - y + 2x + 4 = 0$ as a line concept format $4x + y + 4 = 0$ generates a sequence of procedural scaffoldings. These algebraic procedural scaffoldings apply rules such as the algebraic associative rule, the identity rule and arithmetic calculations in a recursive manner. The geometric conceptual scaffolding internally contains the procedural scaffolding trace. This geometric conceptual scaffolding is attached onto the recognized geometric concept knowledge.

In certain circumstances, the annotation contains the ambiguity to transform as a semantic knowledge. For instance, the input $x = 1$ can be either recognized as a line entity or an attribute. To discriminate it, the context information is required to further check if the literal x already exists. As such conflict happens case by case and depends on the context of existing input, we currently apply the case-based reasoning approach to perform the conflict resolving process [22].

Relation Constraint Checking and Graph Update

The goal of step 2 is to determine if any relation exists between any concepts of DAG and the current semantic concept m. Such relation searching process can be attributed to

a constraint-solving problem (CSP) [29]. Algorithm 1 shows the search procedure upon the semantic DAG.

Data: Current geometric concept m.
Data: Current semantic DAG G.
Result: Detected semantic relations
Part1: unary-relation constraint Search:
for *node* g_i *in* G **do**
  if *Relation_Check*(g_i, m) **then**
    Generate unary-relation.
  end
end
Part2: binary-relation constraint Search:
for *node* g_i *in* G **do**
  for *node* g_j *in* G **do**
    if *Relation_Check*(g_i, g_j, m) **then**
      Generate binary-relation.
    end
  end
end

Algorithm 1: Constraint Checking on DAG.

Relation_Check function determines the existence of relation. We use the case-base reasoning approach to detect relations [22]. Semantic concept types, concept label information all guide the relation detection. When relations are found between current semantic concept m and any concepts on the semantic graph, certain geometric relation procedures will be executed either in a forward or backward reasoning manner. This leads to two types of geometric concept generation: the geometric concept instance generation upon existing DAG nodes or the new geometric concept node generation. The generated instance in the first type will be cached in the existing geometric concept DAG node. Step 3 of the method updates the graph by adding the current geometric concept m and its relation with others geometric concepts on the semantic DAG. If step 2 generates a new geometric concept node, the method in Figure 3 will be recursively called so as to add this geometric concept onto the semantic DAG.

For instance, in the Problem3, after the annotated inputs $A(2, y)$ and $B(-1, 4)$ have been added onto the semantic DAG, we recognize the annotated input $y = -1$ as a geometric attribute in the step 1 of the method. In the step 2, as the label constraint between $y = -1$ and $A(2, y)$ is satisfied, the unary relation between these two geometric concepts is found. The relation procedure to substitute a geometric attribute into a geometric concept is executed. This will generate a geometric concept instance $A(2, -1)$, which is cached onto the existing geometric concept node $A(2, y)$. Step 3 adds the current concept $y = -1$ onto the DAG and build the relation with $A(2, y)$. Similarly, for the next annotated input $d =?$, after recognizing it as a query of geometric distance attribute, we find the binary relation between $d =?$ and $A(2, y)$, $B(-1, 4)$. The binary distance relation procedure is applied, which brings all instances of A and B into the distance function to calculate distance attribute instances. Since there is only one instance A and B respectively, the single distance attribute instance $d = 5$ is generated by substituting the point's X and Y variables into the distance function and further compute the result. The generated $d = 5$ distance instance is attached onto the geometric query node $d =?$.

In addition to the above forward reasoning scenario, the backward reasoning scenario exists, such as Problem7. After parsing annotated inputs $(4, 3), (2, v)$, $A : line, B : line$, $pass - through(A, (4, 3)), pass - through(A, (2, v))$ and $B : m = 1/2$ onto the semantic DAG, recognizing the annotation $perpendicular(A, B)$ as the geometric relation, Step 2 of the method detects the binary relation between it and other two geometric concepts: line A and line B. When the geometric perpendicular relation procedure is executed, we translate the $perpendicular(A, B)$ as the perpendicular utility function $m * m1 = -1$, where $m = 1/2$ has been specified as the line B's slope in another DAG node. The backward reasoning is performed by substituting m's value into the perpendicular utility function, and the new geometric attribute $m1 = -2$ is synthesized. Step 3 further adds this new geometric attribute onto the graph, and triggers the recursive call to the method to detect its relation with other existing nodes on the semantic DAG. As $m1 = -2$ represents the line A's slope, its binary relation with (4,3) and (2,v) is detected recursively and a new geometric attribute $v = 7$ is generated.

In step 2, when a geometric relation procedure is executed, the geometric relation scaffolding will be generated according to the pre-defined instruction templates. For instance, when the distance relation is detected and the distance relation procedure is triggered in the Problem3, a sequence of geometric relation scaffoldings or a scaffolding trace can be generated, such as "substitute a given point into the distance function.", "derive the distance attribute value by manipulating the distance function.". Similar as step 1, the geometric scaffolding may contain algebraic procedural scaffoldings to achieve the goal of geometric scaffolding. In the above distance derivation scaffolding, the algebraic expression manipulation and simplification procedures is required as a linear sequence of algebraic rule derivations.

DAG bottom-up Reification

Step 4 attemps to perform the variable substitution (or reification) using existing relations, and further to propagate this substitution to other concepts through relations. Algorithm 2 shows this bottom-up reification procedure. For instance, in Problem3, reifying the attribute $y = -1$ into the geometric point $A(2, y)$ generates a new cached point instance $A(2, -1)$, which depends on $A(2, y)$. Under some circumstances, the existing DAG can become a cyclic graph. For instance, in Problem7, after the geometrical attribute $v = 7$ is generated in the step 2, a unary relation is detected between it and the geometric concept $(2, v)$. The reification procedure is executed by substituting the value of v into the geometric concept $(2, v)$. The cyclic graph is formed after this relation is added onto this semantic DAG. In order to prevent the infinite constraint propagation, the halt checking process should be conducted so that the propagation can be terminated when the geometric concept node has been visited. Executing the reification procedure will produce the geometric reification scaffolding. For instance, the scaffolding trace of Problem7 contains one geometric scaffolding "Reify the equations internal variable by substituting a given fact.".

Scaffolding Trace Propogation on DAG

Data: Current graph node g_i.
foreach *Graph edge* e_i *in* g_i*'s out edges* **do**
  Graph node $g_j \leftarrow e_i$.target
  if *HaltChecking*(g_j) **then**
    break;
  end
  if *Reify*(g_j, g_i) **then**
    Propagate the reification recursively.
  end
end

Algorithm 2: Reification propogation on the directed graph.

In the automated reasoning (shown in method 3), four steps can generate different types of geometrical scaffoldings, in which each one might contain an algebraic scaffolding trace or a sequence of algebraic scaffolding step. Geometrical scaffolding is attached onto the instance of geometric concept node. When a geometric concept instance is generated, geometrical scaffoldings from other related geometric concepts should be propagated onto this concept instance first. Then the current geometric scaffolding is added onto it. For instance, in Problem7, the scaffolding trace from $m1 = -2$ will be added onto the new generated node $v = 7$. Further geometric relation scaffoldings to derive from $m1 = -2$ to $v = 7$ will be appended after previous scaffoldings onto the $v = 7$. After the reasoning method is completely executed, each geometric concept node may contain a geometric scaffolding trace.

A Step-Wise Tutoring Framework

The generated scaffolding trace is used to construct our model-tracing tutor behind the *AnalyticalInk*. As the scaffolding trace is a linear sequence of geometric scaffoldings, where each of them could contain a linear sequence of algebraic scaffoldings, the scaffolding trace can be seen as a two dimension linear structure. Figure 4 shows this two dimension scaffolding trace structure in a directed graph manner. The outer graph node maps to the geometric concept state and the outer graph edge encodes each geometric scaffolding step to transform one geometric concept from one state to another. A graph node can link to a sequence of inner graph nodes, where each inner node represents the geometric concept's state after applying the algebraic procedural scaffolding. The inner graph edge encodes the algebraic procedural scaffolding. After executing inner algebraic procedures, the inner node directs back to the same outer node. Each directed graph of scaffolding trace maps to a math problem's solving plan. The model-tracing tutor in the *AnalyticalInk* is composed by directed graphs of math problems.

Match-and-Verify User Input Step

Figure 5 shows the method to verify a user's input steps. After parsing the raw user input as a geometric concept m, step 2 attempts to match m with a graph node among the directed graph of scaffolding trace. If the user input does not match any nodes on the scaffolding trace graph, we further evaluate it by adding it on the problems semantic DAG (step 3). This will trigger the automated reasoning method to check its relations with other geometric concepts on the semantic DAG. After processing it, if the auto-reasoning procedure deduces

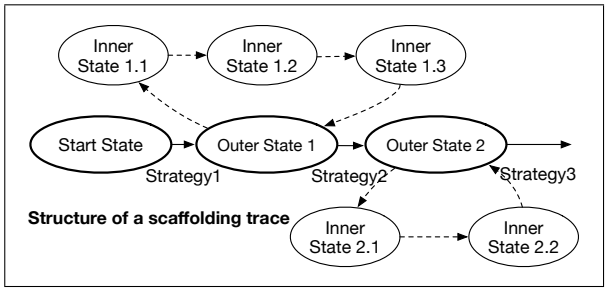


Figure 4: The direct graph structure of a scaffolding trace.

Input: Let s be the user's current input step.

Input: Let G be the semantic DAG representation of a math problem.

Input: Let G' be the scaffolding trace graph of the current problem.

Output: Let v as the boolean value to determine the correctness of s.

Initial: v := false

Step 1: geometric concept m = **Syntax Pattern Match**(s)

where m can be geometric entities, attributes or relations.

Step 2: v = **Match State on Scaffolding Trace Graph**(G', m)

If v : = true return v

Step 3: <v,t> = **Match Concepts on Semantic DAG**(G, m)

where t is a scaffolding trace to transform m to another state

If v := false return v

Step 4: **Update Trace**(G', t)

return true

Figure 5: The hybrid match-and-verify approach to check the user answer.

the user input as a geometric concept with the auxiliary scaffolding trace, we can compare geometric concept states with every state from our automated scaffolding trace. If any two states matched from these two scaffolding traces, it implies that the current user input can direct to the same goal as the automated scaffolding traces derivation, but with a different tracing path. Under this circumstance, the system will update the directed graph of scaffolding trace to insert this new scaffolding trace as different branches (step 4).

Adaptive Scaffolding Selection

According to user inputs, selecting adaptive scaffolding is critical in ITSs [20]. Using scaffolding trace directed graph, we keep a user flag to record the user's current solving status. If the user input is verified as a correct step, the user flag will be directed to the corresponding matched graph node. If the current input is verified as a wrong step, it implies that the system did not match any graph node state. Under this circumstance, the system will utilize the user flag to trace back as the latest correct status of a graph node that the user has been explored. The scaffolding selection is achieved by looking ahead onto the outer edge of current graph node.

AnalyticalInk: Interactive Math Learning Environment

Based on the model-tracing tutor, *AnalyticalInk* learning system keeps track and guide students problem solving by analyzing their interactive inputs. The user interface of system

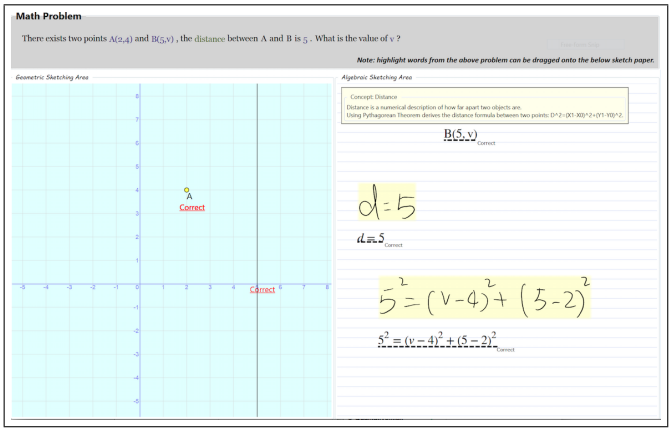


Figure 6: An user scenario where the participant was solving Problem2 using *AnalyticalInk*. The participant dragged $A(2, 4)$ onto the geometric side and $B(5, v)$ onto the algebraic side. The participant draw a line onto the geometric side and sketched $d = 5$ onto the algebraic side. She queried the system and was given the flag feedback. The participant further dragged the distance concept word and the system showed her the "distance" concept explanation. The participant continued to solve the problem.

mainly contains three regions (shown both in Figure 1 and Figure 6). The top region presents the math problem. Below it, there are two working regions: algebraic and geometric work spaces. Scaffoldings are shown as visual widgets onto the system.

Keyword Highlighting and Dragging Interaction

Highlighting keywords of a math problem has been shown to be an effective learning technique in the knowledge acquisition phase [8]. Based on our semantic problem representation, we implement this learning technique and further extend it to allow users drag certain semantic words from the top problem region onto the bottom workspace to get the initial condition or the concept definition from the given problem.

From syntax annotations of a math problem, we further annotate each syntax word to align it to the corresponding phrase in the problem. For instance, in the Figure 6, syntax annotations of the problem can be: $A(2, 4)$, $B(5, v)$, $d = 5$, $v =?$. $A(2, 4)$ and $B(5, v)$ syntax words directly map to the corresponding phrase in the problem. As $d = 5$ is extracted from several problem phrases relations, we annotate it onto the phrase "5" of the problem. We mark the $v =?$ to align with the phrase "v" in the end of problem narratives. Aligned phrases will be color highlighted in the math problem. In addition, these highlight words can be dragged from problem region to either algebraic or geometric region, which depends on the type of geometric concept. In the Figure 6, $A(2, 4)$ is dragged onto the geometric side. As $B(5, v)$ is not a concrete point from the initial condition, so it can be dragged only onto the algebraic side. As the type of the syntax word $d = 5$ is a geometric attribute, this phrase can be dragged onto the algebraic workspace only. Dragging the query phrase from the math problem will visualize geometric conceptual scaffoldings to give students guidance toward the current math problem solving. In Figure 1, dragging the phrase "What" triggers the system to show up all geometric conceptual scaffoldings in the problem solving strategy visualization box. In addition, during the problem acquisition phase, concept glossary explanation is another learning technique to support help seeking [1]. In *AnalyticalInk*, besides syntax word highlighting and dragging, we also annotate certain geometric concept words to be highlighted and draggable. In Figures 6, when dragging the phrase "distance", the system shows the extra explanation of concept distance in the visual widget. These concept explanations are manually annotated into the system.

Geometric Concept and Algebraic Procedural Understanding

AnalyticalInk lets users input the geometric shape concepts, such as a point, a line or a line segment in the geometric workspace, and input algebraic expressions in the algebraic workspace. The system can deduce from a geometric shape to its corresponding algebraic format or vice versa. As quantities in the math problem can have different natural narratives, geometric concept understanding and algebraic procedural derivation need to be iteratively proceeded in order to solve such math problems [28]. For instance, in Figure 6, students need to acquire the geometric point concept, and the distance concept's formula. By deriving the distance formula algebraically, students can finally get the distance geometric concept understanding in the geometric side. Conversely, in Figure 1, students need to acquire the geometric concepts, such as the point, the line slope, lines' perpendicular. Students need to use the slope and the perpendicular's algebraic formula to derive the geometric concept procedurally.

Sketch Input and Gesture Query

Sketch-based cognitive model has been proved to foster the perceptual and conceptual knowledge understanding [10]. Sketching has also shown its effectiveness to facilitate reasoning during math problem solving [25]. *AnalyticalInk* embodies the sketch as its input mode to simulate how students use the pen-and-paper to conduct math problem solving. The system lets users write math expressions, equations on the algebraic canvas, draw geometric shapes and labels on the geometric canvas. The system provides the real-time handwriting recognition feedback as the recognized math typeset offset, which is shown below the hand-drawn strokes. The handwritten math equation recognition is based on StarPad [34]. The geometric shape detection is built by recognizing low-level sketch primitives [27]. Students can further manipulate beautified shapes by touch.

In addition to the sketch input, the system supports the inferential reasoning by letting the user write a question mark as the reasoning tool. *AnalyticalInk* uses the template-matching approach to recognize the handwritten question mark [4]. When the handwritten question mark is recognized, if the written region is near any geometric shapes or algebraic expressions, the system will trigger the match-and-verify procedure. Otherwise, the system will directly show a next-step scaffolding toward the user. By verifying users' input, *AnalyticalInk* provides the flag feedback to indicate the current step's correctness (shown in Figure 1). If the current step is wrong, two scaffolding visualization widgets will be shown on the user interface. The problem solving strategy visual widget on the top of the algebraic canvas presents the linear sequence of geometric concept scaffoldings. One step is highlighted among this sequence to represent the current student problem solving status. The other scaffolding widget shows

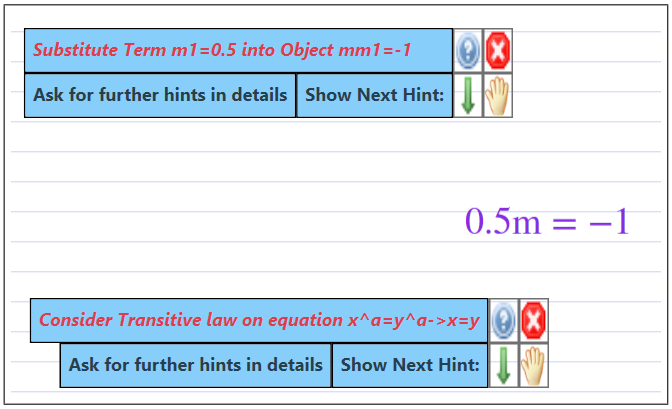


Figure 7: An example shows the algebraic procedural scaffolding derivations. The top didactic scaffolding in the visual widget and the middle math typeset forms one step algebraic procedural scaffolding. Clicking the down arrow on top of the scaffolding visual widget generates another visual widget below to give the next step meta-cognition scaffolding before showing the step's didactic scaffolding, which is similar to the one in the top widget.

the algebraic procedural scaffolding within the current geometric concept scaffolding. Before showing a didactic scaffolding step, the system first shows the corresponding meta-scaffolding or meta-cognition hint to inspire students' self-motived reasoning. Figure 7 displays a meta-cognition hint: *"Consider Transitive law on equation"*. Students can retrieve additional procedural scaffoldings by clicking the down arrow image on the widget.

EVALUATION

AnalyticalInk currently models 15 geometric concepts, which are commonly described in the coordinate geometry section of the geometry course, such as point, line, x-axis, origin, midpoint, line slope, circle, circle radius, length, perpendicular, intersection. Our geometric concept syntax can annotate geometrical mathematical problems. We have used it to annotate 30 problems (Problem1-7 are samples of them). Our current work did not formally evaluate the correctness of reasoning and the scaffolding generation. Instead, two human tutors were recruited to proofread the generated scaffolding trace per problem manually. Both tutors separately showed that the generated scaffolding traces of 30 annotated problems are the correct solutions.

To evaluate the effectiveness of the tutoring framework, learning techniques of *AnalyticalInk*, we performed a human-centered usability evaluation of the current interactive learning prototype system. We performed a with-in subjects study with the keyword highlighting and dragging (with or without) as the independent variable. Both conditions provides the tutoring scaffolding, the dual-canvas problem solving and the natural sketch input. The dependent variables were several aspects of the user events we recorded (see Table 1). This evaluation did not consider the learning effect of the system. R1 considers total user inputs, where R2 and R3 track users' inputs onto the geometric and algebraic canvas. R4 is calculated as the number of times the user makes a question-mark gesture to asks for a hint without input steps. R5 is calculated as the number of times the system writes the same gesture near to user input so as to evaluate it. R6 is calculated as

R1: The number of steps the user inputs.
R2: The number of steps the user inputs on the algebraic side.
R3: The number of steps the user inputs on the geometric side.
R4: The number of times the user makes a query.
R5: The number of times the system evaluates user inputs.
R6: The number of times the system shows the meta scaffolding.
R7: The number of times the system shows the scaffolding.
R8: The number of times the user rectifies an input step.

Table 1: The log file of user event record per problem.

the number of times the system evaluates the user input as wrong. Thus, R5 contains the count of R6. R7 is calculated by counting the number of times the user clicks the down arrow image to retrieve the scaffolding from visual widget. R8 is a counter that records two consecutive queries where the previous query check the user's step as a wrong step, and the current query verifies the user input as a correct step.

We hypothesize that the tutoring framework can help students to check their steps and give appropriate scaffolding feedback in demand. The keyword highlighting and dragging learning technique facilitate their problem understanding. We assume *AnalyticalInk* have the potential to guide students' geometry concept understanding and algebraic procedural skills.

Tasks and Experiment Procedure

We recruited 10 participants (with 9 of age 18 and one of age 17) who were currently taking a college algebra class. The experiment was conducted on a Microsoft Surface 3 Pro. Participants were asked to use the stylus and touch during the experiment. The study asked participants to use *AnalyticalInk* to solve two sets of 10 math word problems (one set with keyword highlighting-dragging and one without). The difficulty level of the problems in the two sets was same. We achieved this by changing numerical values but keeping the problem statement same. The order of conditions was balanced across participants based on the latin-square design.

Since each participant had to solve two sets of similar problems, we asked participants to complete the study in two sessions on consecutive days where each session takes about 70 to 80 minutes to finish. Because of the similarity of problems between two sets, participants might apply the lessons learned in the first session toward the second session, which should be minimized. Though putting one-day delay cannot completely reduce such side effect, it is acceptable for our usability evaluation goal. The problem-solving sessions were video-recorded.

In the first session, participants filled a pre-questionnaire, which collected information about their age, if they have taken the algebra course and their comfort level to solve math problem. They were then introduced to *AnalyticalInk* by letting them use it to solve a sample problem under the guidance of a moderator. The moderator gave instructions on how to use the system, answered participants' queries. Participants then began solving a set of ten problems (with or without keyword highlighting and dragging depending upon the balanced order). The second session was the same as the first session with no pre-questionnaire. During problem-solving sessions, we asked participants talk-aloud to express what they are cur-

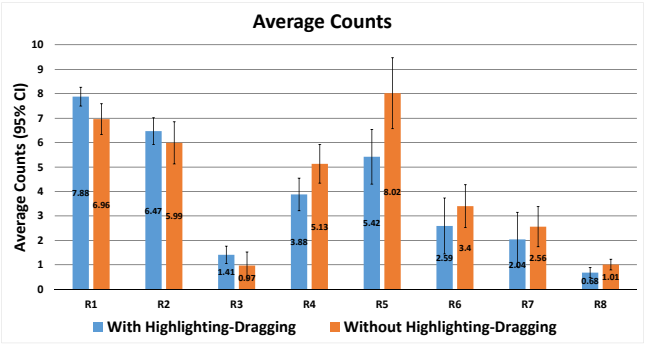


Figure 8: The average mean of each record in the user event log file.

rently doing and what they plan to do for the next action. This allowed the moderator to help participants in case they forget how to use the system to perform certain actions. During the problem solving, participants can explore the learning system to solve each problem. They can perform the keyword dragging, sketch out knowledge input either in the algebraic or the geometric canvas, query the system, and explore the scaffolding trace. When participants tried to verify a written solving step, as the system analyzed recognized expressions instead of participants' written form, participants were instructed to make sure that the recognized typeset expression matches their written form. After completing the two sessions, participants were asked to fill out a questionnaire that required them to rate *AnalyticalInk* on a variety of metrics (shown in Table 2).

Result

From user log files and their questionnaire feedback, we found the evidence that the answer checking of the tutoring framework is useful to guide students' math problem solving. Students prefer to use the keyword highlighting and dragging learning technique. In general, the system is useful to help students to learn geometric concepts and to improve algebraic procedural derivations.

Figure 8 shows the quantitative result from log files under two conditions (with or without the keyword highlighting and dragging learning technique). In both conditions, students either checked their input steps (R5) or asked for a next-step hint based on the last correct step (R4). Students made the error-rectification once in average for each problem solving (R8). In other words, the answer checking in the tutoring framework is effective to help students to correct their own wrong inputs. The users asked for next-step hints (R4) significantly more times without the keyword highlighting and dragging compared to when the technique was present ($t_9 = -2.727, p < 0.05$). The number of times the user input evaluation (R5) took place was significantly higher when the keyword highlighting and dragging was absent ($t_9 = -2.933, p < 0.05$). The number of times the user rectified their input (R8) was significantly less with this learning technique ($t_9 = -2.4, p < 0.05$).

Figure 9 shows the user's qualitative feedback on metrics. Students believed that step verification is useful (Q5). However, students reported that the current geometric concept scaffolding (Q6) and algebraic procedural scaffolding (Q7) is

Keyword Highlighting and Dragging	
Q1	To what extent did the keyword highlighting and dragging consolidate your math concept understanding?
Q2	To what extent did the keyword highlighting and dragging accelerate your math problem-solving?
Q3	How hard did you have to work (mentally and physically) to accomplish your level of performance?
Q4	To what extent did you feel frustrated using this learning technique?
Tutoring Answer Check and Scaffolding	
Q5	How useful was it to verify your solving step?
Q6	How useful and effective was the current geometry concept scaffolding to help your problem solving?
Q7	How useful and effective was the current algebraic procedural scaffolding to help your problem solving?
Overall User Experience	
Q8	How useful and effective was it to let you interact with both the geometric and the algebraic sides?
Q9	How useful and effective was the pen-and-gesture input-and-query flow for the math problem-solving?
Q10	To what extent did you think the current system should let you freely input, edit and query the knowledge?
Q11	How engaged did you feel to use *AnalyticalInk*?
Q12	How useful and effective was it to use *AnalyticalInk*?
Q13	How easy was it to use *AnalyticalInk* for the first time?
Q14	How easy was it to use *AnalyticalInk* after you get familiar with it?
Q15	How satisfied did you feel with this math tutoring system?
Q16	Regarding the keyword highlighting and dragging learning technique, which interaction method would you prefer?

Table 2: Post-Questionnaire. Except for Q16, participants responded to questions on a 7 point likert scale. (1 equals strongly disagree and 7 equals strongly agree).

not that useful as verifying input steps. The user feedback indicated that the keyword highlighting and dragging had a low cognitive load to perform it (Q3-Q4). This learning technique helped them to improve geometric concept understanding and accelerated their math problem solving (Q1-Q2). Nine out of ten participants preferred to use this learning technique during their problem solving (Q16). The participant, who preferred to show original math problem without this learning technique, remembered most of geometric concepts and did not face issues during problem solving. Students agreed that providing both algebraic-and-geometric workspaces is effective for their problem solving (Q8). The log file summary also indicated that students used both algebraic and geometric workspaces to solve math problems (average count of R1, R2 and R3). Students believed that the sketch-based input was effective for math problem solving (Q9). Overall, students gave the positive feedback (Q11-Q12) on the effectiveness and usefulness of *AnalyticalInk*. In general, they were satisfied with the current prototype system (Q15).

DISCUSSION AND FUTURE WORK

The current *AnalyticalInk* can handle a limited set of annotated geometry math problems. As part of this paper, we did not systematically evaluate the annotation syntax of geometric concepts and corresponding semantic. Future work should

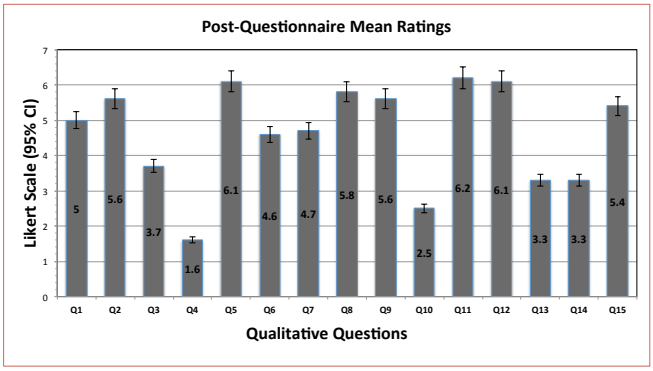


Figure 9: Mean participant ratings of individual aspects of *AnalyticalInk*'s user interface. Except for Q16, in Q3, Q4, 1 was the most positive answer and 7 was the most negative answer; in other questions, 1 was the most negative answer and 7 was the most positive answer.

evaluate its variability to handle large-scale math geometry word problems.

Step-wise Tutoring Framework: For the answer checking of the tutoring framework, though participants gave the positive feedback, we observed that the system sometimes did make the false positive matching-and-verification errors. Variations of written formats are attribute to such error. For instance, one participant sketches "x+2+4+1=x+2+5=x+7", which was recognized as a wrong step. However it was correct if the expression can be divided as two algebraic equations. Future work should handle more written formats. Participants informed they were unsatisfied with the current geometric concept and algebraic procedural scaffoldings. We summarized three reasons based on our observation and user feedback. First, current scaffolding narratives follow the rigorous rule-based description to derive geometric concepts. But participants commented out that verbalized example-based narratives should also be present so as to explain applied rules intuitively. Second, current geometric concept scaffoldings are insufficient for students who do not master the concept. The current geometric concept scaffolding covers the simple description to describe the process to transform to the corresponding algebraic procedurals. Rich geometric concept level pedagogical scaffoldings are required before showing the algebraic procedural scaffolding. Third, the adaptive scaffolding selection is not accurate. When the system recognizes the current user step as a wrong step, the system shows the scaffolding based on the last correct step. However, the current user step might slip several rule-based steps, in which the user might only made one rule deduction error. Under such circumstance, the current system cannot skip these correctly applied rule steps and show the error deduction scaffolding accurately. Future work should refine the scaffolding narrative and augment pedagogical concept-based scaffoldings. Annotated error-based scaffolding should be built into our model-tracing tutor. Based on such improvements, the adaptive scaffolding evaluation should be conducted to measure efforts and outcomes of the system toward learning [12].

Keyword highlighting and dragging: The log file quantitative data implied that students asked for less hints and checked less input steps with this learning technique (R4 and R5). R8 showed that students rectified wrong steps few times with this technique. Our observation indicated that with this learning technique, students spent more time to understand the initial condition of a math problem. By highlighting keywords of a problem, students increased their engagement and substantially built the solving scheme or a plan. Dragging such semantic words onto the workspace reinforced their planning. By dragging geometric concept words, they can retrieve and memorize explanations to facilitate the later problem solving. We believed this learning technique increased students' certainty to input correct steps, which indirectly reduced the times to query the system(R4), verify steps (R5).

User experience of the system: Though participants rated sketch-and-gesture flow to be effective to facilitate their problem solving. However, they reported that the system still has constraints to let them freely input, edit and query the knowledge (Q10). From our observation, participants faced sketch recognition errors during problem solving and recognition errors hindered the solving process. We gave participants pre-instructions to minimize making the recognition errors before the study. For a written step, if the participant did not want to query it, she can ignore the recognition error if it exists, and further derived the next step. However, if the participant wanted to query a step, the recognition result for the current step must be matched with her written intention. When learning to use the system, participants reported that it was not easy to use the system for the first time (Q13). Such perception existed even they got familiar with the system (Q14). By observing their actions during the study, though the cognitive load of sketching are low, sketch recognition errors affected their reasoning. It makes participants feel that the system is not easy to use even when they get familiar with it. Further sketch recognition improvement should be investigated.

CONCLUSION

In this paper, we present *AnalyticalInk*, an interactive math learning environment to facilitate geometry math problem solving. The graph representation of geometry math problems was illustrated. Based on it, we showed the automated reasoning method to solve these problems and generate instructional scaffoldings. When tutoring students to solve such problems, we described a hybrid match-and-verification framework to verify users' solving steps. The user interface of *AnalyticalInk* highlights keywords of a problem, and further supports the keyword dragging to the geometry-and-algebra workspaces. The system lets students use both geometry-and-algebra workspaces to understand geometrical concepts and derive algebraic procedures in an iterative manner. We conducted a usability evaluation of the system. Our results showed that the user step verification was useful. The keyword highlighting and dragging learning technique helped users to understand a problem and plan to solve it. The sketch input and gesture query supported the natural input and reasoning during the math problem solving. Overall, users were satisfied with the current *AnalyticalInk*.

ACKNOWLEDGMENTS

This work is supported in part by NSF CAREER award IIS-0845921. We would like to thank Fei Liu to revise the math problem semantic representation. We would like to thank the members of ISUE lab for their support and the anonymous reviewers for their useful comments and feedback.

REFERENCES

1. Aleven, V., McLaren, B., Roll, I., and Koedinger, K. Toward tutoring help seeking. In *Intelligent Tutoring Systems*, J. Lester, R. Vicari, and F. Paraguau, Eds., vol. 3220 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, 227–239.
2. Aleven, V., Mclaren, B. M., Sewall, J., and Koedinger, K. R. A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education 19*, 2 (2009), 105–154.
3. Anderson, J. R., Corbett, A. T., Koedinger, K. R., and Pelletier, R. Cognitive tutors: Lessons learned. *The journal of the learning sciences 4*, 2 (1995), 167–207.
4. Anthony, L., and Wobbrock, J. O. A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010*, GI '10, Canadian Information Processing Society (Toronto, Ont., Canada, Canada, 2010), 245–252.
5. Anthony, L., Yang, J., and Koedinger, K. R. A paradigm for handwriting-based intelligent tutors. *Int. J. Hum.-Comput. Stud. 70*, 11 (Nov. 2012), 866–887.
6. Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse* (August 2013).
7. Corbett, A. T., and Anderson, J. R. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction 4*, 4 (1994), 253–278.
8. Dunlosky, J., Rawson, K. A., Marsh, E. J., Nathan, M. J., and Willingham, D. T. Improving students learning with effective learning techniques promising directions from cognitive and educational psychology. *Psychological Science in the Public Interest 14*, 1 (2013), 4–58.
9. Fillmore, C. J. *The need for a frame semantics in linguistics*. Scriptor, 1977.
10. Forbus, K. D., Ferguson, R. W., and Usher, J. M. Towards a computational model of sketching. In *Proceedings of the 6th international conference on Intelligent user interfaces*, ACM (2001), 77–83.
11. Ford, B. Parsing expression grammars: a recognition-based syntactic foundation. In *ACM SIGPLAN Notices*, vol. 39, ACM (2004), 111–122.
12. González-Brenes, J. P., and Huang, Y. Your model is predictivebut is it useful? theoretical and empirical considerations of a new paradigm for adaptive tutoring evaluation. In *Proceedings of the 8th Intl. Conf. on Educational Data Mining* (2015).
13. Gulwani, S., Korthikanti, V. A., and Tiwari, A. Synthesizing geometry constructions. In *ACM SIGPLAN Notices*, vol. 46, ACM (2011), 50–61.
14. Heffernan, N. T., and Koedinger, K. R. The composition effect in symbolizing: The role of symbol production vs. text comprehension. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates Hillsdale, NJ (1997), 307–312.
15. Heffernan, N. T., Koedinger, K. R., and Razzaq, L. Expanding the model-tracing architecture: A 3rd generation intelligent tutor for algebra symbolization. *International Journal of Artificial Intelligence in Education 18*, 2 (2008), 153.
16. Ilany, B.-S., Margolin, B., et al. Language and mathematics: Bridging between natural language and mathematical language in solving problems in mathematics. *Creative Education 1*, 03 (2010), 138.
17. Jiang, Y., Tian, F., Wang, H., Zhang, X., Wang, X., and Dai, G. Intelligent understanding of handwritten geometry theorem proving. In *Proceedings of the 15th international conference on Intelligent user interfaces*, ACM (2010), 119–128.
18. Koedinger, K. R., Anderson, J. R., Hadley, W. H., and Mark, M. A. Intelligent tutoring goes to school in the big city.
19. Koedinger, K. R., Booth, J. L., and Klahr, D. Instructional complexity and the science to constrain it. *Science 342*, 6161 (2013), 935–937.
20. Koedinger, K. R., Brunskill, E., Baker, R. S., McLaughlin, E. A., and Stamper, J. New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine 34*, 3 (2013), 27–41.
21. Koedinger, K. R., and Nathan, M. J. The real story behind story problems: Effects of representations on quantitative reasoning. *The Journal of the Learning Sciences 13*, 2 (2004), 129–164.
22. Kolodner, J. *Case-based reasoning*. Morgan Kaufmann, 2014.
23. Kushman, N., Artzi, Y., Zettlemoyer, L., and Barzilay, R. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics (Baltimore, Maryland, June 2014), 271–281.
24. LaViola, Jr., J. J., and Zeleznik, R. C. Mathpad2: A system for the creation and exploration of mathematical sketches. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, ACM (New York, NY, USA, 2004), 432–440.
25. Oviatt, S. Interfaces for thinkers: Computer input capabilities that support inferential reasoning. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, ICMI '13, ACM (New York, NY, USA, 2013), 221–228.
26. Paquette, L., Lebeau, J.-F., Beaulieu, G., and Mayers, A. Designing a knowledge representation approach for the generation of pedagogical interventions by mtts. *International Journal of Artificial Intelligence in Education 25*, 1 (2015), 118–156.

27. Paulson, B., and Hammond, T. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, ACM (2008), 1–10.

28. Rittle-Johnson, B., Siegler, R. S., and Alibali, M. W. Developing conceptual understanding and procedural skill in mathematics: An iterative process. *Journal of educational psychology 93*, 2 (2001), 346.

29. Russell, S., and Norvig, P. Artificial intelligence: a modern approach.

30. Schoenfeld, A. H. Learning to think mathematically: Problem solving, metacognition, and sense making in mathematics. *Handbook of research on mathematics teaching and learning* (1992), 334–370.

31. Seo, M. J., Hajishirzi, H., Farhadi, A., Etzioni, O., and Malcolm, C. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015* (2015), 1466–1476.

32. Vanlehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M. The andes physics tutoring system: Lessons learned. *Int. J. Artif. Intell. Ed. 15*, 3 (Aug. 2005), 147–204.

33. Zeleznik, R., Bragdon, A., Adeputra, F., and Ko, H.-S. Hands-on math: A page-based multi-touch and pen desktop for technical work and problem solving. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, ACM (New York, NY, USA, 2010), 17–26.

34. Zeleznik, R., Miller, T., Li, C., and Laviola Jr, J. J. Mathpaper: Mathematical sketching with fluid support for interactive computation. In *Smart Graphics*, Springer (2008), 20–32.

35. Zhou, L., Dai, S., and Chen, L. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics (Lisbon, Portugal, September 2015), 817–822.