A Framework for Automated Generation of Questions Across Formal Domains

Rahul Singhal, Martin Henz, and Shubham Goyal

School of Computing
National University of Singapore
Singapore
{rahulsinghal,dcsmjh,idmsg}@nus.edu.sg

Abstract. In this work, questions are tasks posed to secondary students to help them understand a subject, or to help educators assess their level of competency in it. Automated question generation is important today as content providers in education try to scale their efforts. In particular, MOOCs need a continuous supply of new questions in order to offer educational content to thousands of students, and to provide a fair assessment process. This paper innovates in three ways; (1) we describe question generation across domains, and, previous efforts as instances of a general framework; (2) we establish first-order logic as a suitable formal tool to describe question scenarios, questions and answers; and (3) we generalize a published question generation method based on logic programming and theorem proving to work across domains. We apply this approach to three domains in high school education—geometry, algebra and mechanics (physics)—and report initial results.

Keywords: First-order logic, automated deduction, pattern matching, formal domains, axiomatic approach, Constraint Handling Rules (CHR)

1 Introduction

Developing assessment material is laborious. Teachers spend countless hours scavenging textbooks and developing original exercises for practice worksheets, homework problems, remedial material, and exams. To prevent cheating, many teachers write several versions of each test, multiplying the work required for creating problems. Various standardized tests such as GRE, SAT and GMAT regularly require new questions. In addition, motivated students would like to have access to a large number of questions for practice.

This demand has become more acute through the rising popularity of massive open online course (MOOCs), in which tens of thousands of students may be enrolled in the same course. This massive scale poses a significant technical challenge: creating a large and diverse set of problems of varying difficulty, preventing cheating, and providing new practice problems to students. Hence, there is a need of a software that can quickly generate a large number of questions.

The best developed tutoring systems in the domains under investigation—JGEX [Gao and Lin, 2004], Geogebra [Hohenwarter et al., 2010] and Cinderella [Cin,

2013] for geometry, ActiveMath [Melis and Siekmann, 2004] for algebra, and Andes [Vanlehn et al., 2005] for physics—do not yet generate questions automatically. The published approaches for generating questions—[Singh et al., 2012] and [Alvin et al., 2014]—describe how to do so based on given *similar* questions.

In this paper, we address the problem of automatically generating new questions and solutions to the satisfaction of a user, based on a specified topic in a given, formally described domain. The proposed framework generates a *scenario* from the user-specified input and then repeatedly adds automatically deduced or generated consistent information, until a question is generated that satisfies the user's requirements.

In order to generalize our previous work on question generation to handle new domains, first-order logic emerges as the tool of choice. We start from a decscription of *domains* as first-order structures in Section 3, and include high school geometry, algebra and physics as examples. The first task in question generation consists of generating a scenario from the user input. Section 4 describes scenarios as first-order formulas. Section 5 shows how several question formats can be characterized as sets of formulas with specific shape. In Section 6, we describe our question generation framework as a process of generating scenario formulas followed by their continuous refinement. Finally, we present preliminary results in Section 7.

2 Related Work

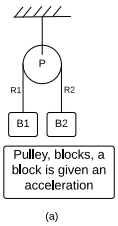
A question generation methodology for high school algebra is proposed in Singh et al. [2012], which generates questions similar to a given one, based on a combination of synthesizing and numerical techniques. The generated questions in this approach resemble the given question syntactically and the approach does not consider solution generation.

Alvin et al. [2014] propose an algorithm for generating geometry proof questions for a high school curriculum. The problem generation approach firstly generates a hypergraph that represents all possible proofs over a given pair of user-provided figures and axioms. Later, it systematically enumerates all possible goal sets to find interesting problems. However, the approach is semi-automated as the user needs to provide a figure to generate questions.

Singhal et al. [2014a] propose a framework for automated generation of high school geometry questions. In addition, Singhal et al. [2014b], Singhal et al. [2014c] extend the framework for generation of questions requiring implicit construction and handling congruent regions. However, the presentation is specific to the geometry domain.

Current research is domain-specific and mainly focuses on generating *similar* questions based on the user-given question. To the best of our knowledge, there is no such general framework which can generate questions for multiple domains from the user-desired domain-specific inputs.

Figure 1 Figure explaining question in mechanics domain. (a) is the pictorial representation of the generated/predefined scenario (b)actual representation of the scenario in the framework (c) shows an instance of Question-setting C (d) instance of a generated fact (e) dependency of acceleration of block (f) a new fact generated using the dependencies.



(a)

Scenario requirements	Predicate and signatures	Scenario requirements	Predicate and signatures
A stationary pulley	stationary_pulley (P)	Block B1 is connected with rope R1	connected (B1, R1)
A block	block (B1),	Rope R1 is connected with rope R2	connected(R1, R2)
A rope	rope(R1)	Force by gravity on block B1 with mass m1	force_downward_gravity (B1, m1g)
mass of block B1 is M1	mass_block(B1, M1)	acceleration of block B1 is the result	acc_upward_block (B1, (M2-M1)/(M1+ M2)*G)

(b)

	Initial state	Final State
Initialization step	User input (null, null, null)	Figure 1a, 1b (G, null, null)
Extension step	Figure 1a, 1b (G, H= null, null)	Hn = Generating facts related to equality of forces (G, H U Hn, null)
Enrichment step	(G, H , null)	Gn = Instantiating masses of blocks (G U Gn , H, null)
Commitment step	(G, H U Q, null)	(G, H U Q, Q)

(c)

Generated fact	Axiom used
equal(T1, T2)	connected(P, R1), connected(P, R2), connected(R1, R2), rope(R1), rope(R2), stationary_pulley(P), massless_pulley(P), force(R1, T1), force(R2, T2) :- equal(T1, T2).

(d)

dependency(B_1, A_1) -> mass_block(B_1, m_1), mass_block(B_2, m_2)

(e)

acc_block(B_1, A_1) = (m_2- m_1)/(m_1+ m_2)*G

(f)

3 Domains

A domain consists of objects, relationships between the objects and the rules regarding their arrangement. A domain theory can be represented in a first order logic where each domain-object can be described by a unary predicate and relationship between the objects can be represented by non-unary predicates. Each predicate has a signature where each signature is a set of predicate symbols, each with a given arity. Arity is a finite list of sorts, e.g. $[X_s] = [X_1, X_1, ..., X_n]$.

A domain rule/concept can be represented in axiomatic form where each axiom is a conjunction of horn clause. A domain-rule basically is the description of generation of a particular relationship between the objects under a specified arrangement of objects. The arrangement becomes the head in the axiomatic format and the generated relationships represents the body. For explanation, we have used three different domains- geometry, algebra and physics.

3.1 Physics domain

In physics domain, various domain-objects such as pulley, rope, blocks can be represented by the unary predicates. Figure 1b shows some examples of predicates and signatures of Figure 1a. Relationships between the objects can be described by the predicates. *connected*(P, R) is a predicate having arity of sorts such as pulley, rope. In addition, a sort can be of numeric type such as real numbers useful for representing the quantitative relationships. For example, Force-

block(B, 10) describes the force acting on the block is 10 unit. Figure 1d shows an example of domain-rule in mechanics sub-domain. The rule shows that the two forces, "T1" and "T2", acted on the blocks are equal if the pulley is massless.

3.2 Algebra domain

In algebra domain, we are referring the work proposed by Singh et al. [2012] and trying to represent the questions in the first-order logic. In this paper, each *term* can be represented by a predicate. Figure 3b shows one predefined-scenario given by the user in trigonometry domain. Figure 3a shows a scenario representation in the predicate form. The domain rules will involve the axioms for testing the generated question by replacing the free variables with different numeric values. Figure 3h shows a rule for testing the generated question.

3.3 Geometry domain

In geometry domain, objects such as triangle, line and points can be represented by n-ary predicates such as point(X), line(X,Y) and triangle(X,Y,Z) respectively. The detail of geometry-domain representation is provided here Singhal et al. [2014a].

Predicates are being created for all the domain-specific types and their properties. The choice of representing the objects and their attributes of a domain in a predicate form is subjective. Hence, the framework allows the user to define his own predicates. Depending on the defined predicate, the user needs to define domain-specific rules and predefined scenarios. The predicates will be used in the other components with the same semantic definition. The framework will generate questions according to the user-defined predicates and domain-specific rules/concepts.

3.4 Implementation

SWI-Prolog (Version 7.1.2) Schrijvers and Demoen [2004] is used for implementation of predicates. The axioms are represented using Constraint Handling Rules (CHR) Frühwirth and Raiser [2011]. In our implementation, we use the CHR library provided by K.U.Leuven, on top of SWI-Prolog. A domain may require solving of linear equations solver. SWI-Prolog library of CLPR is used for solving them.

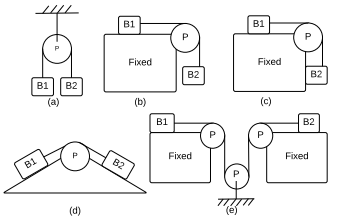
4 Scenarios

In first-order logic, a scenario is a formula which can be represented as

$\exists \bar{x}\ \phi(\bar{x})$, where $\bar{x}$ represents a list of variables.

Scenario is the arrangement of domain-specific objects in the user-defined way. A scenario is a set of facts representing the objects, their attributes and the known relationships between them. A fact consists of the predicate whose

Figure 2 Figure showing new scenarios from an existing one with the help of rules. (a) Shows an exiting figure scenario. (b) is generated by from (a) by adding a fixed wedge. (c) is generated from (a) by adding a fixed wedge and attaching both the blocks with it. (d) is generated from (a) by adding an inclined wedge. (e) is generated from (a) by adding multiple wedges, pulleys and blocks (f) solution of the generated question in mechanics domain.



G_k	Axioms	H_k
Figure 1a, 1b, 1c	Force diagram on block B1	$T_1 - m_1g = m_1A_1$
	Force diagram on block B2	$m_2g - T_2 = m_2A_2$
	Equal force acting by the ropes	$T_1 = T_2$
	Equal acceleration of blocks	$A_1 = A_2$
	Linear equation solver	$A_1 = (m_2-m_1)*g/(m_1+ m_2)$

(f)

structure has been mentioned in section 3. A scenario is considered as a question with holes which can be filled depending on the user's requirement. A scenario is either generated from the user-defined rules for adding/removing/modifying the objects or from the predefined-arrangement. For explanation, we have used three different domains- geometry, algebra and mechanics(physics).

In Physics domian, the user has defined the rules for adding pulleys, blocks, ropes and wedges in a given scenario. In addition, Figure 2a shows the pictorial representation of a predefined-scenario. From Figure 2a, various new scenarios have been generated by applying user-defined rules for adding wedges and pulley. Figure 2b,c,d,e shows the pictorial representation of various new generated scenarios. Figure 1c shows the partial scenario in the form of facts for Figure 2a.

In Algebra domain, the paper is generating the question similar to what is given by the user. Hence, for our framework, these questions will wb taken as predefined scenarios. Figure 3a,b shows the scenario representation which is a collection of facts.

In Geometry domain, the user has defined the rules for adding domain-specific objects such as perpendicular, median, angle-bisector in a given scenario. The detail of geometry-domain representation is provided here Singhal et al. [2014a].

5 Questions

In first-order logic, a question is the addition of information on the top of a scenario. It may be represented in three different schemes:

1. $\exists\bar{x}\ (\phi(\bar{x}) \wedge \varphi(\bar{x}))$, where $\varphi(\bar{x})$ is a question for finding the relationship φ.
2. $\exists\bar{y}\ \exists\bar{x}\ (\phi(\bar{x}) \wedge (\varphi(\bar{x}, y_i) \wedge \varphi(\bar{x}, y_j) \rightarrow y_i = y_j))$, where the question is finding y_i.
3. $\exists\bar{y}\ \exists\bar{z}\ \exists\bar{x}\ (\phi(\bar{x}) \wedge (\varphi(y_i, z_j) \wedge \varphi(y_k, b_l) \rightarrow y_i = y_k \wedge z_j = z_l))$, where the question is finding y_i and z_j.

A question Q generated by our system can be represented by an ordered tuple, (scenario S_G, question-fact P) where: scenario S_G is mentioned in Section 4. P

refers to the fact that can be represented as a question. For example, Figure 1f can be converted as to a question which involves finding the acceleration of the block B1.

We propose a process of state transformation to generate questions where each state is of type question-setting C, which can be described as : C = (G, H, Q), where G $\subseteq$ initial facts, H $\subseteq$ Facts derived from the initial facts with the help of predefined axioms, Q $\subseteq$ Facts which can be used as a new question. Both P and G are subset of S_G. However, H may not always be subset of S_G.

Each item in the Question-setting C changes along the process of question-generation. The various steps in the question generation process are as follows:

1. Initialization step, C_{is} = ($\varnothing$, $\varnothing$, $\varnothing$) $\rightarrow$ (G, $\varnothing$, $\varnothing$)
2. Extension step, C_{ex} = (G, H, $\varnothing$) $\rightarrow$ (G, H $\cup$ H_n, $\varnothing$)
3. Enrichment step, C_{en} = (G, H, $\varnothing$) $\rightarrow$ (G $\cup$ G_n, H , $\varnothing$)
4. Commitment step, C_{cs} = (G, H $\cup$ Q, $\varnothing$) $\rightarrow$ (G, H $\cup$ Q, Q)

Figure 1c shows the change in C for the mechanics domain. C_{is} refers to the generation of facts from the user input. C_{ex} refers to the generation of new facts from the predefined axioms and initial facts. One example of C_{ex} is the generation of the fact H_n that the forces acting on the ropes are equal, which does not satisfies the user requirement. C_{en} refers to the scenario enrichment by instantiating some facts G_n. C_{cs} refers to the generation of the fact that can be used as a question such as related to the acceleration of the blocks.

In terms of first-order logic, the axioms and solution can be represented as:

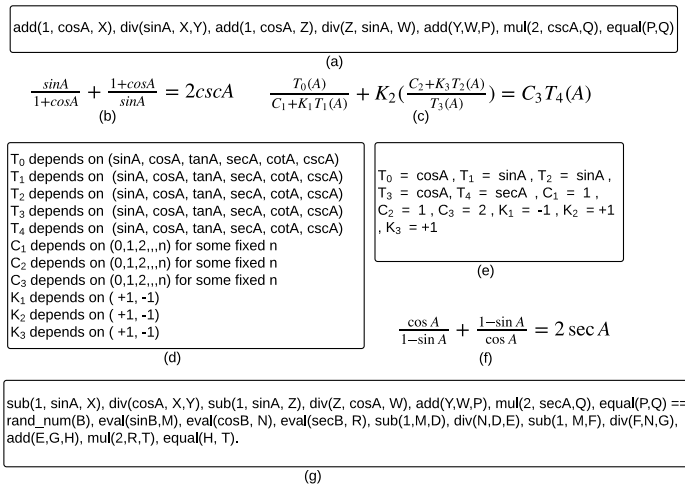
1. $\exists \bar{x}\ (\phi(\bar{x}) \wedge \Phi(\bar{x}) \wedge \varphi(\bar{x}))$, where $\Phi(\bar{x})$ shows the axioms used and $\varphi(\bar{x})$ is the answer.
2. $\forall \bar{y}\ \forall \bar{z}\ \exists \bar{x}\ (\phi(\bar{x}) \wedge \Phi(\bar{x}) \wedge (\varphi(\bar{x}, y_i) \wedge \varphi(x, y_j) \rightarrow y_i = y_j)\ \wedge y_i = c)$, where $y_i = c$ shows the answer.
3. $\exists \bar{y}\ \exists \bar{z}\ \exists \bar{x}\ (\phi(\bar{x}) \wedge \Phi(\bar{x}) \wedge (\varphi(y_i, z_j) \wedge \varphi(y_k, b_l) \rightarrow y_i = y_k \wedge z_j = z_l) \wedge y_i \wedge z_j)$, where $y_i \wedge z_j$ is a answer.

The solution of the generated question refers to the steps that lead to the answer. It can be generated by tracing the facts and the axioms which leads to the generation of the question fact from the initial facts. Mathematically, the solution can be represented as

(G_k , axioms) $\rightarrow$ H_k , (H_k, axioms) $\rightarrow$ Q,
where $G_k \subseteq G \cup G_n$, $H_k \subseteq H \cup Q$
(G_k , axioms) refers to the application of axioms on G_k
Figure 2f shows a solution for generating the acceleration of block B1.
We have proposed a claim regarding the suitable domains for our system.

Claim for the suitability of the domain
Any domain, where the knowledge can be represented in the form of *scenarios* and the domain concepts/rules can be represented in the *axiomatic format*, the questions along with the solutions can be generated from that domain.

Figure 3 Figure (a),(b) shows the actual and pictorial representation of the scenario and (c) shows the representation after application of rules (d) is the generated dependencies of the variables used in the scenario (e) shows an instance of selected dependencies (f) shows the pictorial representation of the generated scenario from the selected dependencies (g) shows the rule written in CHR form for testing the correctness of the generated question.



6 Framework

Our framework comprises of three major components along with the knowledge databases used for storing scenarios and a set of predefined rules. The framework detail can be found in our previous work Singhal et al. [2014a].

The input given by the user is fed into the first component, *Generating scenario (GS)*. This component generates a set of scenarios S_o from the input. Figure 1a and b shows the pictorial and the actual representation of the generated scenario from the input given by the user. If no scenario is given, the system generates a scenario according to the user-defined ways. The resulting scenario is passed and modified by the other components.

The scenario is passed to the second component, *Generating Facts and Solutions (GFS)*. This component is used to find values of the unknown variables with the help of predefined axioms. The new values can be considered an answer to a question that can be generated. Figure 1d shows the generated new fact about the forces acting due to the ropes. One of the possible question generated from this fact can be finding the relationship between the forces acting due to ropes. The detail of the algorithm used by the component can be found in Singhal et al. [2014a]. In addition, some domains involve solving of linear equations for generation of newly generated facts. Hence, CLPR (a linear equation solver) is integrated in this component. For example, Singhal et al. [2014c] proposed additional algorithm to generate new facts.

If the suitability conditions for the generated scenario S_o are not met then the scenario S_o is fed into the last component, *Scenario Enrichment (SE)*. SE enriches the scenario via an algorithm which is used for generating sets of facts which needs to be instantiated for scenario enrichment. For example, Figure 1e shows that for obtaining the acceleration of a block B1, masses of blocks B1 and B2 need to be instantiated.

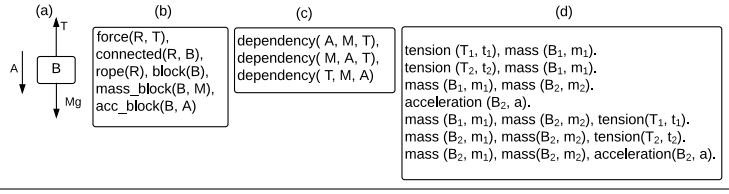
The generated scenario from this component is again passed to GFS component to get the new facts (question) and this loop continues until a question meeting suitability condition is found. Figure 1f shows the value of acceleration of a block computed by using one instance of the acceleration dependencies. The generated questions would involve finding the acceleration of each block, given their masses.

6.1 Knowledge database for predefined domain-specific rules (KR)

KR is a knowledge database of the domain-specific rules/concepts in the form of axioms. Constraint handling rules (CHR) is used for the representation as it is suited more to our requirement. The domain-specific scenarios and rules may require multiple goals to be satisfied simultaneously for pattern matching. CHR has incremental and concurrent properties where multiple goals (predicates) can be tested. In contrast to Prolog, the rules are multi-headed and are executed in a committed-choice manner using a forward chaining algorithm.

In addition to rules for generating new facts, KR have rules for finding the dependency of the variables used in the scenario. The dependency facts D generated by KR are used by the SE component for the scenario enrichment. The head of these rules represents the rules/concepts to be matched and the body stores the dependencies of the variables used in the head. For example, Figure 4c shows that dependency of acceleration of the block depends on the mass of the block and force acting on the block from the rope. Dependency of a variable helps the system to know the variables need to be instantiated to get the dependent variable. For example, to get acceleration of the block, predicates for mass of the block and force by rope needs to be instantiated.

Figure 4 Figure explaining the rules in KR in physics. (a) shows the pictorial representation of the pattern inside the scenario (b) shows representation of scenario (c) shows generated dependencies of the variables based on the KR rules (d) shows the list of dependencies of the acceleration of block B1 in terms of other variables present in the current scenario.



6.2 Generating algebra questions as a special case

User-given question is considered as a predefined scenario. The scenario is acted by the domain-specific user-defined rules to impose constraints (relational and functional). The rules are written in Prolog and generates a general problem from the given scenario. Figure 3c shows the generated general set of problems obtained by application of rules. Figure 3d shows the dependencies generated from the user-defined rules. At this moment, GFS component would not be able to generate any new fact as no free variables are instantiated. Hence, GD component will instantiate free variables with some values. Figure 3e,f shows an example of instantiation and generation of a new question after substitution of the values. GGFS component will now check the correctness of the generated question by assigning some values to both sides and ensuring the equality. Figure 3g shows a rule written in CHR form for equality testing.

7 Correctness and Experimental Results

The framework firstly generates a scenario from the given user input. A scenario is a formula which holds true. In rest of the process, the framework is making the hidden information explicit.

The proposed system can generate questions in multiple domains using the framework described in section 6. The generated questions depend on the predefined knowledge database which varies with the domain. We generated question in various domains such as high school 2D geometry and mechanics(physics). For each domain we have a different number of predefined axioms, for example (50 axioms in geometry and 20 axioms in physics).The time taken to generate questions in each domain is less than 10 sec.

8 Conclusion

In this paper, we provide a framework for the automatic generation of questions for multiple domains. We showed existing approaches as special case. We proposed a notion for describing a domain to generate user-desired questions and solutions and described a promising tool for our framework.

Future work can be carried in various directions. One major drawback of this approach would be the inability of the framework to handle degeneracy conditions across domains. For example, the framework cannot check the correctness of the function for adding a pulley to a given scenario. Other improvements would involve conducting an experiment in which the teachers would be asked to differentiate between the system generated problems from existing online and textbook questions.

Bibliography

Cinderella geometry tool (Jan 2013), `http://www.cinderella.de/tiki-index.php`

Alvin, C., Gulwani, S., Majumdar, R., Mukhopadhyay, S.: Synthesis of geometry proof problems. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence. pp. 245–252 (2014)

Frühwirth, T., Raiser, F. (eds.): Constraint Handling Rules: Compilation, Execution, and Analysis. Books On Demand (2011)

Gao, X.S., Lin, Q.: MMP/Geometer—a software package for automated geometric reasoning. In: Winkler, F. (ed.) Automated Deduction in Geometry, pp. 44–66. No. 2930 in Lecture Notes in Artificial Intelligence, Springer (2004)

Hohenwarter, J., Hohenwarter, M., Lavicza, Z.: Evaluating difficulty levels of dynamic mathematics software tools to enhance teachers' professional development with dynamic mathematics software. Journal for Technology in Mathematics Education 17(3), 127–134 (2010)

Melis, E., Siekmann, J.: ActiveMath: An intelligent tutoring system for mathematics. In: et al., L.R. (ed.) Artificial Intelligence and Soft Computing—ICAISC 2004, pp. 91–101. No. 3070 in Lecture Notes in Artificial Intelligence, Springer (2004)

Schrijvers, T., Demoen, B.: The K. U. Leuven CHR System—implementation and application. In: First Workshop on Constraint Handling Rules—Selected Contributions. pp. 1–5 (2004)

Singh, R., Gulwani, S., Rajamani, S.: Automatically generating algebra problems. In: Brodley, C.E., Stone, P. (eds.) Proceedings of the 26th AAAI Conference on Artificial Intelligence. pp. 1620–1627 (2012)

Singhal, R., Henz, M., McGee, K.: Automated generation of geometry questions for high school mathematics. In: Proceeedings of the Sixth International Conference on Computer Supported Education (2014a)

Singhal, R., Henz, M., McGee, K.: Automated generation of geometry questions for high school mathematics involving implicit construction. In: Proceeedings of the Sixth International Conference on Computer Supported Education (2014b)

Singhal, R., Henz, M., McGee, K.: Automated generation of region based geometry questions. In: Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence (2014c)

Vanlehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R.H., Taylor, L., Treacy, D.J., Weinstein, A., Wintersgill, M.C.: The Andes physics tutoring system—five years of evaluations. In: Proceedings of the Artificial Intelligence in Education Conference. pp. 678–685 (2005)