A method for automated generation of exercises with similar level of complexity

João Mendes[1], **João Marcos**[2]

[1]Programa de Sistemas e Computação – PPgSC
Universidade Federal do Rio Grande do Norte (UFRN) – Natal – RN – Brazil

[2]Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN) – Natal – RN – Brazil

`mendeslopes.joao@gmail.com, jmarcos@dimap.ufrn.br`

Abstract. *The effort put by an educator on the manual construction of questions may be reduced if one uses tools for the automated generation of questions. Among these tools, only very few are able to control the level of complexity of their output, which ends up representing a challenge for the dissemination of individualized assessments. In the present study, we propose a method for the automated generation of exercises which uses the structure of the solution for the conjecture given as input in order to guarantee the similarity in the level of complexity of these exercises.*

1. Introduction

The activity of manually constructing exercises requires a prodigious expenditure of time in an educator's professional routine. Such effort may end up subtracting time from other important tasks connected to the students' learning process, such as the development of individualized assessments. Aiming to tackle this problem, Automatic Question Generation (AQG) techniques came up. In the context of AQG, a question is any statement (not only those ending with a question mark) that might be used as a pedagogical assignment.

Despite the increase in the development of AQG techniques in the recent days, there is a historical and substantial lack of AQG techniques with controlled level of complexity [Kurdi et al. 2020]. Not being able to manage this aspect of automatically generated questions, for instance, harm some students in assessments in which they receive questions of a higher level of complexity than they should be prepared to answer.

In order to implement AQG techniques with controlled level of complexity, some approaches have already been proposed in the literature. In [Wang and Su 2016], the metric used to define the level of complexity of the Mathematical Word Problems is based on the number of arithmetic operations in the equation that represents the problem. In contrast, in [Singhal et al. 2016], the amount of objects in the scenario of the problem concerning the question, for example, the amount of blocks and pulleys in a mechanics problem, is one of the factors taken into account in measuring the associated complexity. In our work, we propose a method with a proof-theoretical ground, based on what we call "proof-schemas", to generate exercises with a similar level of complexity.

2. Pedagogical benefits

One effective way of mitigating plagiarism in assessments is by individualizing them [Manoharan 2017]. Doing this can help increasing the guarantees such assessments really attest the intended learning goals are achieved. Moreover, this format of assessment aids in the management of adjustments on the applied level of demand since it provides feedback on the development of the learning process of each student.

To make the individualization of assessments a feasible practice, the requirement of more time for creating a number of questions larger than or equal to the amount of students is a critical barrier [Millar and Manoharan 2021]. Such barrier, we claim, can be overcome by a tool implemented following the method presented here.

3. The big picture of the method

To express the complexity of the generated exercises, our method lies in a notion of minimality of proofs. A proof P for a conjecture φ is minimal if there is no other proof for φ that is less complex than P. The complexity of a proof, in turn, is determined by the amount of the theory-specif rules applied. Do note that one conjecture might have more than one minimal proof.

Initially, we might develop an appropriate proof system having rules written in a way that make the polarities explicit (namely, representing the judgments of assertion and denial, as applied to statements of the underlying language) of their antecedents and succedents. Then, given a hand-curated conjecture, our process of generation has basically two steps:

1. Proof-schemas construction: Using an appropriate proof system, we construct proof-schemas based on minimal proofs of the input conjecture. The proof-schema is a tree that shows in an abstract format a possible way of verifying the validity or refutability of the conjecture, depending on whether it is a valid conjecture or not.
2. Search for conjectures with similar level of complexity: Based on the rules of the same proof system, we search for all the conjectures with derivation trees that are isomorphic to some of the previously constructed proof-schemas.

In the end of the process, illustrated by the diagram in Figure 1, we obtain a set of different conjectures (the "exercises" we refer to) that can be proved using the same proof-schemas.

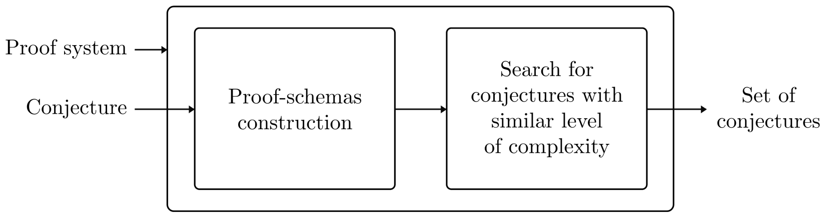

Proof system
Conjecture
Proof-schemas construction
Search for conjectures with similar level of complexity
Set of conjectures

Figure 1. The big picture of the process of automated generation.

The minimality of the proofs used to construct the proof-schemas in the first step is fundamental because, depending on the proof system, proofs having different complexities might be constructed given one conjecture as input. This possibility would 1. allow us to generate exercises that are, actually, more or less complex than the exercise given as input and 2. prevent us to compare the complexity between exercises generated by different inputs.

4. An example using a fragment of the set-theoretic language

For this example, consider a first-order theory in which the underlying language, L_{ST}, contains connectives to represent set-theoretic operations and relations. In this language, there is no symbol for negation. To compensate for this, every formula is preceded by either a $+$ or a $-$. For any formula φ of the underlying language, whenever we write $+\ \varphi$ we mean "the assertion of φ" and whenever we write $-\ \varphi$ we mean "the denial of φ". The importance of this choice will be clarified in the explanation of the second step of our method. Futhermore, to construct some of the atomic assertions of such language, we also have, as a primitive symbol, $\in$ representing the membership relation.

Associated to the deductive system used in the present example, the language contains the symbols $\cup$, $\cap$, $\setminus$, $\ominus$, $\subseteq$ and $)\!($. On the operational side, $\cup$, $\cap$, $\setminus$ and $\ominus$ represent, respectively, union, intersection, set difference and symmetric difference. On the relational side, $\subseteq$ and $)\!($ represent, respectively, subset and disjointness (the latter consists in the relation that holds between any pair of sets having an empty intersection).

The subset and disjointness relations are both defined in terms of universal quantifications on the elements of the sets they relate. That's why, we need, in the rules for introduction of these relations, an assertion to declare the name of a constant representing an arbitrary element of the domain of the theory. The assertion $\mathtt{Set}\ c$ plays exactly such role.

Now, consider a proof system with rules described below:

Pure logical rule

$$\dfrac{\begin{array}{cc}\vdots & \vdots\\ \varphi & \tilde{\varphi}\end{array}}{\divideontimes}\ \mathbf{PPS}(\varphi)$$

Inclusion

$$\begin{array}{c}\dfrac{}{\mathtt{Set}\ \ c}\,[i] \qquad \dfrac{}{+\ c \in c_1}\,[j] \qquad \dfrac{}{-\ c \in c_2}\,[k] \\ \vdots \\ \dfrac{\divideontimes}{+\ c_1 \subseteq c_2}\ +\!\subseteq\!\mathbf{I}[i,j,k]\end{array}$$

Disjointness

$$\begin{array}{c}\dfrac{}{\mathtt{Set}\ \ c}\,[i] \qquad \dfrac{}{+\ c \in c_1}\,[j] \qquad \dfrac{}{+\ c \in c_2}\,[k] \\ \vdots \\ \dfrac{\divideontimes}{+\ c_1\ )\!(\ c_2}\ +)\!(\mathbf{I}[i,j,k]\end{array}$$

Union

$$\dfrac{\begin{array}{cc}\vdots & \vdots\\ +\ c_0 \in c_1 \cup c_2 & -\ c_0 \in c_1\end{array}}{+\ c_0 \in c_2}\ +\!\cup\!\mathbf{E}_1 \qquad \dfrac{\begin{array}{cc}\vdots & \vdots\\ +\ c_0 \in c_1 \cup c_2 & -\ c_0 \in c_2\end{array}}{+\ c_0 \in c_1}\ +\!\cup\!\mathbf{E}_2$$

$$\dfrac{\begin{array}{c}\vdots\\ -\ c_0 \in c_1 \cup c_2\end{array}}{-\ c_0 \in c_2}\ -\!\cup\!\mathbf{E}_1 \qquad \dfrac{\begin{array}{c}\vdots\\ -\ c_0 \in c_1 \cup c_2\end{array}}{-\ c_0 \in c_1}\ -\!\cup\!\mathbf{E}_2$$

Intersection

$$\dfrac{\begin{array}{c}\vdots\\ +\ c_0 \in c_1 \cap c_2\end{array}}{+\ c_0 \in c_2}\ +\!\cap\!\mathbf{E}_1 \qquad \dfrac{\begin{array}{c}\vdots\\ +\ c_0 \in c_1 \cap c_2\end{array}}{+\ c_0 \in c_1}\ +\!\cap\!\mathbf{E}_2$$

$$\dfrac{\overset{\vdots}{-\; c_0 \in c_1 \cap c_2} \qquad \overset{\vdots}{+\; c_0 \in c_1}}{-\; c_0 \in c_2}\; -\cap\mathbf{E}_1 \qquad\qquad \dfrac{\overset{\vdots}{-\; c_0 \in c_1 \cap c_2} \qquad \overset{\vdots}{+\; c_0 \in c_2}}{-\; c_0 \in c_1}\; -\cap\mathbf{E}_2$$

Set difference

$$\dfrac{\overset{\vdots}{+\; c_0 \in c_1 \setminus c_2}}{-\; c_0 \in c_2}\; +\setminus\mathbf{E}_1 \qquad\qquad \dfrac{\overset{\vdots}{+\; c_0 \in c_1 \setminus c_2}}{+\; c_0 \in c_1}\; +\setminus\mathbf{E}_2$$

$$\dfrac{\overset{\vdots}{-\; c_0 \in c_1 \setminus c_2} \qquad \overset{\vdots}{+\; c_0 \in c_1}}{+\; c_0 \in c_2}\; -\setminus\mathbf{E}_1 \qquad\qquad \dfrac{\overset{\vdots}{-\; c_0 \in c_1 \setminus c_2} \qquad \overset{\vdots}{-\; c_0 \in c_2}}{-\; c_0 \in c_1}\; -\setminus\mathbf{E}_2$$

Symmetric difference

$$\dfrac{\overset{\vdots}{+\; c_0 \in c_1 \ominus c_2} \qquad \overset{\vdots}{+\; c_0 \in c_1}}{-\; c_0 \in c_2}\; +\ominus\mathbf{E}_{1a} \qquad \dfrac{\overset{\vdots}{+\; c_0 \in c_1 \ominus c_2} \qquad \overset{\vdots}{+\; c_0 \in c_2}}{-\; c_0 \in c_1}\; +\ominus\mathbf{E}_{2a}$$

$$\dfrac{\overset{\vdots}{+\; c_0 \in c_1 \ominus c_2} \qquad \overset{\vdots}{-\; c_0 \in c_1}}{+\; c_0 \in c_2}\; +\ominus\mathbf{E}_{1b} \qquad \dfrac{\overset{\vdots}{+\; c_0 \in c_1 \ominus c_2} \qquad \overset{\vdots}{-\; c_0 \in c_2}}{+\; c_0 \in c_1}\; +\ominus\mathbf{E}_{2b}$$

$$\dfrac{\overset{\vdots}{-\; c_0 \in c_1 \ominus c_2} \qquad \overset{\vdots}{+\; c_0 \in c_1}}{+\; c_0 \in c_2}\; -\ominus\mathbf{E}_{1a} \qquad \dfrac{\overset{\vdots}{-\; c_0 \in c_1 \ominus c_2} \qquad \overset{\vdots}{+\; c_0 \in c_2}}{+\; c_0 \in c_1}\; -\ominus\mathbf{E}_{2a}$$

$$\dfrac{\overset{\vdots}{-\; c_0 \in c_1 \ominus c_2} \qquad \overset{\vdots}{-\; c_0 \in c_1}}{-\; c_0 \in c_2}\; -\ominus\mathbf{E}_{1b} \qquad \dfrac{\overset{\vdots}{-\; c_0 \in c_1 \ominus c_2} \qquad \overset{\vdots}{-\; c_0 \in c_2}}{-\; c_0 \in c_1}\; -\ominus\mathbf{E}_{2b}$$

Some observations regarding this proof system:

- The overline tilde has the role of inverting the polarity of the formula in which it is applied, that is, we have $\widetilde{+\;\varphi} := -\;\varphi$ and $\widetilde{-\;\varphi} := +\;\varphi$ for any formula φ of the underlying language.
- $\divideontimes$ is a symbol to represent an absurd.
- In all binary rules, the main formula is assumed to belong to the root of the left-hand derivation tree. This is will bring an important advantage during the second step of our method.
- The polarity in the label of a rule refers to the polarity of its succedent.

  For this example, the conjecture given as input is $+\; A \cap (B \cup C) \subseteq (A \cap B) \cup C$.

4.1. Proof-schemas construction

The derivability of $+\; A \cap (B \cup C) \subseteq (A \cap B) \cup C$ with our proof system shall, then, be proved by one of the following derivation trees:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{-\; c \in (A \cap B) \cup C}\,k}{-\; c \in A \cap B}\; -\cup\mathbf{E}_2 \qquad \dfrac{\dfrac{\dfrac{}{+\; c \in A \cap (B \cup C)}\,j}{+\; c \in B \cup C}\; +\cap\mathbf{E}_1 \quad \dfrac{\dfrac{}{-\; c \in (A \cap B) \cup C}\,k}{-\; c \in C}\; -\cup\mathbf{E}_1}{+\; c \in B}\; +\cup\mathbf{E}_2}{-\; c \in A}\; -\cap\mathbf{E}_2 \qquad \dfrac{\dfrac{}{+\; c \in A \cap (B \cup C)}\,j}{+\; c \in A}\; +\cap\mathbf{E}_2}{\dfrac{\divideontimes}{+\; A \cap (B \cup C) \subseteq (A \cap B) \cup C}\; +\subseteq\mathbf{I}[i, j, k]}\; \mathbf{PPS}(-\; c \in A)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{-\ c\in(A\cap B)\cup C}\,{}^{k}}{-\ c\in A\cap B}\,{-\ \cup\mathbf{E}_2} \quad \dfrac{\dfrac{}{+\ c\in A\cap(B\cup C)}\,{}^{j}}{+\ c\in A}\,{+\ \cap\mathbf{E}_2}}{-\ c\in B}\,{-\ \cap\mathbf{E}_1} \quad \dfrac{\dfrac{\dfrac{}{+\ c\in A\cap(B\cup C)}\,{}^{j}}{+\ c\in B\cup C}\,{+\ \cap\mathbf{E}_1} \quad \dfrac{\dfrac{}{-\ c\in(A\cap B)\cup C}\,{}^{k}}{-\ c\in C}\,{-\ \cup\mathbf{E}_1}}{+\ c\in B}\,{+\ \cup\mathbf{E}_2}}{\divideontimes}\,\mathbf{PPS}(-\ c\in B)}{+\ A\cap(B\cup C)\subseteq(A\cap B)\cup C}\,{+\subseteq\mathbf{I}[i,j,k]}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{-\ c\in(A\cap B)\cup C}\,{}^{k}}{-\ c\in C}\,{-\ \cup\mathbf{E}_1} \quad \dfrac{\dfrac{\dfrac{}{+\ c\in A\cap(B\cup C)}\,{}^{j}}{+\ c\in B\cup C}\,{+\ \cap\mathbf{E}_1} \quad \dfrac{\dfrac{\dfrac{}{-\ c\in(A\cap B)\cup C}\,{}^{k}}{-\ c\in A\cap B}\,{-\ \cup\mathbf{E}_2} \quad \dfrac{\dfrac{}{+\ c\in A\cap(B\cup C)}\,{}^{j}}{+\ c\in A}\,{+\ \cap\mathbf{E}_2}}{-\ c\in B}\,{-\ \cap\mathbf{E}_1}}{+\ c\in C}\,{+\ \cup\mathbf{E}_1}}{\divideontimes}\,\mathbf{PPS}(-\ c\in C)}{+\ A\cap(B\cup C)\subseteq(A\cap B)\cup C}\,{+\subseteq\mathbf{I}[i,j,k]}$$

Despite all three proofs have the same complexity, the choice of the tree might affect the output. Then, we are going to show how a proof-schema for each one of the three derivations can be constructed. With a minimal derivation tree in hand, the proof-schema is constructed by doing the following replacements in this order:

1. Except for the root node and antecedents/succedent of **PPS**, all the nodes are replaced by the placeholder $\bigcirc$.
2. In the root node, the first set-theoretic operation of the input conjecture is replaced by $\circ_1$, the second by $\circ_2$, and so on. The set-theoretic relation, in turn, is replaced by $\leftrightsquigarrow$. So, in the root of the proof-schema, we have $+\ A\circ_1(B\circ_2 C)\leftrightsquigarrow(A\circ_3 B)\circ_4 C$. The result of removing its polarity is what we call the "conjecture-schema" of this proof-schema and this specific conjecture-schema is going to be referred in the remaining of the text as `CS`. Moreover, the set $\{\circ_1,\circ_2,\circ_3,\circ_4,\leftrightsquigarrow\}$ is said to be the set of "meta-connectives", which are symbols representing connectives of the underlying language.
3. Regarding the theory-specific rules, we need to replace, in their labels, the symbols of polarities and the connectives they are either introducing or eliminating:
   - The polarities are replaced by a $\square$. Since the polarity of the root node is not omitted in the proof-schema and the polarity in the label of a rule of our proof-system refers to the polarity of its succedent, there is no need to replace the polarity of the label of the rule that has the root node as the succedent.
   - Taking the representation of the conjecture-schema into account, the connectives are replaced by either $\leftrightsquigarrow$ or $\circ_n$, where $n\in\{1,2,3,4\}$ in this case.

   After this, we obtain what we call the "rules-schema labels" of our proof-schema. The rule $-\ \cup\mathbf{E}_2$, for instance, is replaced by the rule-schema label $\square\circ_4\mathbf{E}_2$.
4. The argument of **PPS** rule is replaced by $\textcircled{S}$A in the first derivation (actually, it could be by any new fresh symbol), $\textcircled{S}$B in the second and by $\textcircled{S}$C in the third. Hence, the antecedents of **PPS** in both derivations must be replaced by $\textcircled{S}$X and $\widetilde{\textcircled{S}}$X as the specification of the rule states, where $X\in\{A,B,C\}$.

The result of the construction of the proof-schemas is shown below:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\bigcirc}\,{}^{k}}{\bigcirc}\,\square\circ_4\mathbf{E}_2 \quad \dfrac{\dfrac{\dfrac{}{\bigcirc}\,{}^{j}}{\bigcirc}\,\square\circ_1\mathbf{E}_1 \quad \dfrac{\dfrac{}{\bigcirc}\,{}^{k}}{\bigcirc}\,\square\circ_4\mathbf{E}_1}{\bigcirc}\,\square\circ_2\mathbf{E}_2}{\textcircled{S}\mathrm{A}}\,\square\circ_3\mathbf{E}_2 \quad \dfrac{\dfrac{}{\bigcirc}\,{}^{j}}{\widetilde{\textcircled{S}\mathrm{A}}}\,\square\circ_1\mathbf{E}_2}{\divideontimes}\,\mathbf{PPS}(\textcircled{S}\mathrm{A})$$
$$\dfrac{\divideontimes}{+\ A\circ_1(B\circ_2 C)\leftrightsquigarrow(A\circ_3 B)\circ_4 C}\,{+\leftrightsquigarrow\mathbf{I}[i,j,k]}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{\bigcirc}^{\,k}}{\bigcirc}\,\square\circ_4\mathbf{E}_2 \quad \dfrac{\overline{\bigcirc}^{\,j}}{\bigcirc}\,\square\circ_1\mathbf{E}_2}{\circledS\mathbf{B}}\,\square\circ_3\mathbf{E}_1 \qquad \dfrac{\dfrac{\overline{\bigcirc}^{\,j}}{\bigcirc}\,\square\circ_1\mathbf{E}_1 \quad \dfrac{\overline{\bigcirc}^{\,k}}{\bigcirc}\,\square\circ_4\mathbf{E}_1}{\widetilde{\circledS\mathbf{B}}}\,\square\circ_2\mathbf{E}_2}{\divideontimes}\,\mathbf{PPS}(\circledS\mathbf{B})}{{\color{blue}+}\ A\circ_1(B\circ_2 C)\leftrightsquigarrow(A\circ_3 B)\circ_4 C}\,{+}{\leftrightsquigarrow}\mathbf{I}[i,j,k]$$

$$\dfrac{\dfrac{\dfrac{\overline{\bigcirc}^{\,k}}{\circledS\mathbf{C}}\,\square\circ_4\mathbf{E}_1 \qquad \dfrac{\dfrac{\overline{\bigcirc}^{\,j}}{\bigcirc}\,\square\circ_1\mathbf{E}_1 \quad \dfrac{\dfrac{\overline{\bigcirc}^{\,k}}{\bigcirc}\,\square\circ_4\mathbf{E}_2 \quad \dfrac{\overline{\bigcirc}^{\,j}}{\bigcirc}\,\square\circ_1\mathbf{E}_2}{\bigcirc}\,\square\circ_3\mathbf{E}_1}{\widetilde{\circledS\mathbf{C}}}\,\square\circ_2\mathbf{E}_1}{\divideontimes}\,\mathbf{PPS}(\circledS\mathbf{C})}{{\color{blue}+}\ A\circ_1(B\circ_2 C)\leftrightsquigarrow(A\circ_3 B)\circ_4 C}\,{+}{\leftrightsquigarrow}\mathbf{I}[i,j,k]$$

The process of a proof-schema construction described here applies for the theory we are working and the proof system we are using in this example. Depending on the theory and the pure logical rules the proof system has, some adjustments and additions in the replacement method need to be done. For example, a proof using a rule for the principle of excluded middle (if we had it in the proof system) would demand a replacement that take into consideration the hypothesis discharged by this rule.

4.2. The search for conjectures with similar level of complexity

Recall that, based on the rules of the proof system presented in 4, this step consists in search for all the conjectures that have a similar level of complexity to the one given as input. For any formula of L_{ST} with the same syntactical structure of $\texttt{CS}$, the way we do this is by verifying if it is possible to recover a derivation for the assertion (or for the denial if the conjecture given as input was a denial) of such formula that is isomorphic to some of the previously constructed proof-schemas in the proof-schemas construction step.

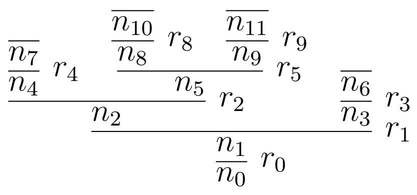
Let φ be a formula of L_{ST} with the same syntactical structure of $\texttt{CS}$ and $\texttt{PS}$ a proof-schema constructed in the proof-schemas construction step. To do the previously mentioned verification, we check, for any rule-schema label r of $\texttt{PS}$, if it is possible to substitute r for some label l of a rule of our proof system. If it is possible, we do this substitution and, based on the rule that l refers to, we recover the content of its antecedent(s) and succedent. Otherwise, we stop the verification with φ and restart the process with another formula of the underlying language.

The checking before the substitution of a rule-schema label is done by taking into consideration the rules of the connective that the meta-connective in the rule-schema label represents. It is not possible, for example, to substitute the rule-schema label $\square\circ_1\mathbf{E}_1$ for the label of any rule about $\ominus$, because $\square\circ_1\mathbf{E}_1$ represents a unary rule and all the rules about $\ominus$ in our proof system are binary.

The order of the checkings before the substitution of a rule-schema label is determined by the breadth-first search traversal of the meta-connectives in the abstract syntactical representation of $\texttt{CS}$. Finally, in case we manage to traverse the whole proof-schema, then ${\color{blue}+}\ \varphi$ has a similar level of complexity to the one of ${\color{blue}+}\ A\cap(B\cup C)\subseteq(A\cap B)\cup C$.

In addition, we actually need to take only the polarities of the antecedents and succedents of the rules into consideration when checking if it is possible to substitute a rule-schema label by a rule label of our proof system. As it is going to be explicit in the example, this is a consequence of having all binary rules with the main formula belonging to the root of the left-hand derivation tree.

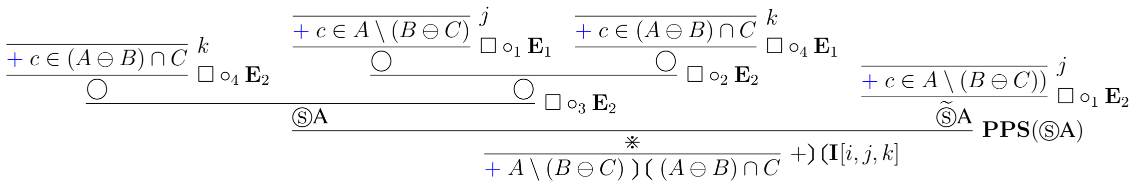
As an example, we are going to use now only the first constructed proof-schema. So as to facilitate the reference to the nodes of the proof-schema, we label them in a breadth-first search traversal. Then, n_0 represents the root, n_1, represents the node containing ✻, n_2 represents the node containing ⓈA, n_3 represents the node containing $\widetilde{\text{Ⓢ}\text{A}}$, and so on. A rule-schema label r_i, in turn, is the one that has the node n_i as its succedent. The tree containing this labeling is presented below:



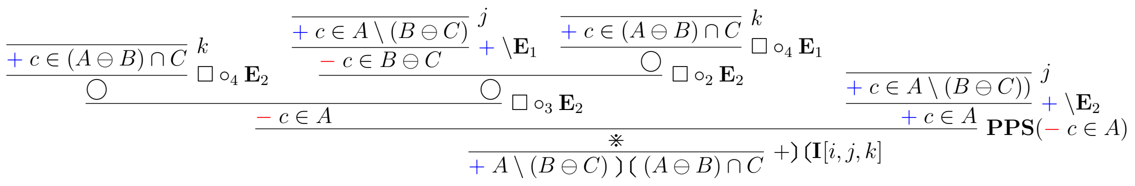
Considering the abstract syntactical representation of CS, the breadth-first search traversal of the meta-connectives is $\leftrightsquigarrow$, $\circ_1$, $\circ_4$, $\circ_2$, $\circ_3$. Thus, the order of the attempt of substitutions of the rules-schema labels in the proof-schema is r_0, r_8, r_3, r_4, r_9, r_5, r_2. Since the meta-connectives in r_3 and r_8 are the same, the attempt of substituting them can be done simultaneously. The same applies to r_4 and r_9.

Let's check, now, if the conjecture $+\ A \setminus (B \ominus C)\ )\!(\ (A \ominus B) \cap C$ is in the output of this example, that is, if it has a similar level of complexity when compared to the input, $+\ A \cap (B \cup C) \subseteq (A \cap B) \cup C$. In this case, $\circ_1$, $\circ_2$, $\circ_3$, $\circ_4$, $\leftrightsquigarrow$ represent, respectively, $\setminus$, $\ominus$, $\ominus$, $\cap$ and $)\!($.

The first attempt of substitution to be done is in the rule-schema label r_0. After this, we might recover the content of the nodes n_7, n_{10}, n_{11} and n_6:



We continue by trying to substitute the rules-schema labels r_8 and r_3. Do note the substitution of the latter allows us to recover the content of the node n_3. As a consequence, we are able recover also the argument of **PPS** and, hence, the content of the node n_2:



Repeating the same process to r_4, r_9, r_5 and r_2, a derivation tree that proves the conjecture $+\ A \setminus (B \ominus C)\ )\!(\ (A \ominus B) \cap C$ and that is isomorphic to the proof-schema is, finally, recovered:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{+\, c \in (A \ominus B) \cap C}^{\,k}}{+\, c \in A \ominus B}{+\cap\mathbf{E}_2} \qquad \dfrac{\dfrac{\overline{+\, c \in A \setminus (B \ominus C)}^{\,j}}{-\, c \in B \ominus C}{+\setminus\mathbf{E}_1} \quad \dfrac{\overline{+\, c \in (A \ominus B) \cap C}^{\,k}}{+\, c \in C}{+\cap\mathbf{E}_1}}{+\, c \in B}{-\ominus\mathbf{E}_{2a}}}{-\, c \in A}{+\ominus\mathbf{E}_{2a}} \qquad \dfrac{\overline{+\, c \in A \setminus (B \ominus C))}^{\,j}}{+\, c \in A}{+\setminus\mathbf{E}_2}}{\divideontimes}\mathbf{PPS}(-\, c \in A)}{+\, A \setminus (B \ominus C) \mathbin{)\!(} (A \ominus B) \cap C}{+\mathbin{)\!(}\mathbf{I}[i,j,k]}$$

Thus, as result of this example, we generate the following conjectures as output $+\ A \cap (B \cup C) \subseteq (A \cap B) \cup C$, $+\ A \cap (B \ominus C) \subseteq (A \cap B) \cup C$, $+\ A \setminus (B \setminus C) \subseteq (A \setminus B) \cup C$, $+\ A \setminus (B \ominus C) \subseteq (A \setminus B) \cup C$, $+\ A \setminus (B \setminus C) \subseteq (A \ominus B) \cup C$, $+\ A \setminus (B \ominus C) \subseteq (A \ominus B) \cup C$, $+\ A \cap (B \cup C) \mathbin{)\!(} (A \ominus B) \setminus C$, $+\ A \cap (B \ominus C) \mathbin{)\!(} (A \ominus B) \setminus C$ and $+\ A \setminus (B \ominus C) \mathbin{)\!(} (A \ominus B) \cap C$. In this case, the output is the same if we repeat this search process to the other two proof-schemas, that is, the choice of the derivation tree in the proof-schemas construction step would have not affected the output.

5. Conclusion and future works

In this work, we presented a method for the automated generation of exercises with similar levels of complexity, illustrating how it works with an example from Set Theory. As an immediate future work, we intend to develop a computational tool that implements the method presented here. Moreover, in order to mechanize the process of constructing minimal derivation trees, we intend to investigate questions regarding Complexity Theory by exploring the literature on cut-based tableaux [D'Agostino 1999].

6. Acknowledgments

The student author is funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and, during the development of this work, he was partially funded by a fellowship from the Group of Tutorial Education Program of Computer Science of Federal University of Rio Grande do Norte (PETCC-UFRN), supported by the Education Development National Fund (FNDE).

References

D'Agostino, M. (1999). *Tableau Methods for Classical Propositional Logic*, pages 45–123. Springer Netherlands, Dordrecht.

Kurdi, G., Leo, J., Parsia, B., Sattler, U., and Al-Emari, S. (2020). A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30(1):121–204.

Manoharan, S. (2017). Personalized assessment as a means to mitigate plagiarism. *IEEE Transactions on Education*, 60(2):112–119.

Millar, R. and Manoharan, S. (2021). Repeat individualized assessment using isomorphic questions: a novel approach to increase peer discussion and learning. *International Journal of Educational Technology in Higher Education*, 18(1):22.

Singhal, R., Goyal, S., and Henz, M. (2016). User-defined difficulty levels for automated question generation. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 828–835.

Wang, K. and Su, Z. (2016). Dimensionally guided synthesis of mathematical word problems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2661–2668. AAAI Press.