Metadata of the chapter that will be visualized in SpringerLink

Book Title	ICT Education	
Series Title		
Chapter Title	Towards an Automated Assistant for Generating Mathematics Problems	
Copyright Year	2022	
Copyright HolderName	Springer Nature Switzerland AG	
Corresponding Author	Family Name	**Mahlaza**
	Particle	
	Given Name	**Zola**
	Prefix	
	Suffix	
	Role	
	Division	Department of Informatics
	Organization	University of Pretoria
	Address	Pretoria, South Africa
	Email	z.mahlaza@up.ac.za
	ORCID	http://orcid.org/0000-0001-9829-1480
Abstract	High school learners in low-income countries are negatively impacted by the SARS Covid-19 pandemic. Consequently, university lecturers of mathematics will have to offer remedial classes to bridge the gap. Since they cannot create practice problems at scale for each student's needs, there is a need for computational tools to do so. There are no existing and published formalisations of mathematical problems that abide by the South African curriculum to allow the automatic generation of problems. We aim to address this gap by formalising exam problems written by grade 12 South African learners in the period 2008–2020. We evaluate the problem types by demonstrating 65% coverage of the 74 matric rewrite problems from the years 2011–2018. The presented problem formalisations allow the generation of maths problems to be used for student-led remedial practice.	
Keywords (separated by '-')	Mathematics education - Mathematical formalisation - Controlled natural language - Natural language generation - Digital educational tools	

Towards an Automated Assistant for Generating Mathematics Problems

Zola Mahlaza(✉)

Department of Informatics, University of Pretoria, Pretoria, South Africa
z.mahlaza@up.ac.za

Abstract. High school learners in low-income countries are negatively impacted by the SARS Covid-19 pandemic. Consequently, university lecturers of mathematics will have to offer remedial classes to bridge the gap. Since they cannot create practice problems at scale for each student's needs, there is a need for computational tools to do so. There are no existing and published formalisations of mathematical problems that abide by the South African curriculum to allow the automatic generation of problems. We aim to address this gap by formalising exam problems written by grade 12 South African learners in the period 2008–2020. We evaluate the problem types by demonstrating 65% coverage of the 74 matric rewrite problems from the years 2011–2018. The presented problem formalisations allow the generation of maths problems to be used for student-led remedial practice. AQ1

Keywords: Mathematics education · Mathematical formalisation · Controlled natural language · Natural language generation · Digital educational tools

1 Introduction

Secondary education was negatively impacted in low-income countries after governments announced lockdowns in response to the Covid-19 pandemic. When the South African government announced relaxations to its first lockdown, the department of Basic Education considered three models for re-opening schools to avoid losing the entire year:

1. "two separate sets of teachers and pupils use the same school building [but] one set [uses] in the morning [and another set] in the afternoon" [8]
2. "groups/grades of learners alternate classes/lessons on different days of the week" [8]
3. "bi-weekly rotational attendance" [8]

All these considered models required that learners spend less time at school and may require additional effort from high school teachers. This is likely to lead to cohorts of first-year university learners who do not have the proper mathematics

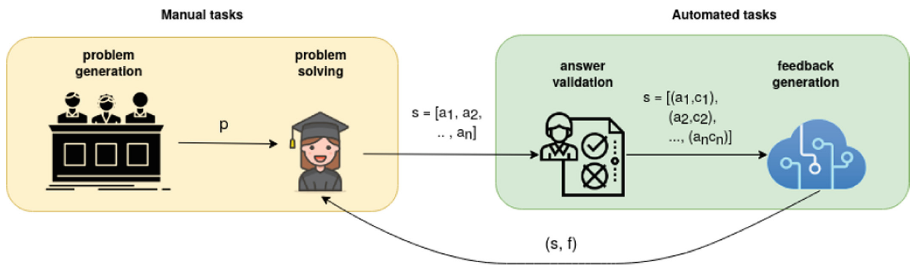

Fig. 1. Existing framework for validating and feedback provision. Abbreviations and symbols used: p = problem, a_i = answer step i, c_i = correctness of answer step i, and f = feedback.

foundation. Consequently, university lecturers will have to offer remedial classes. A more attractive solution is the development and introduction of computer-based education technologies to assist university lecturers.

Existing technology can already be used by university lectures to assist learners in accordance with the framework depicted in Fig. 1. In this framework, an educator(s) manually creates a maths problem ($\mathbf{p}$) and when a student is given the problem, they produce a solution ($\mathbf{s}$) that has multiple steps ($\{\mathbf{a_1}, ..., \mathbf{a_n}\}$). The validity of each step can be automatically checked by an answer validation service and the validated answer ($\{(\mathbf{a_1}, \mathbf{c_1}), ..., (\mathbf{a_n}, \mathbf{c_n})\}$), where $\mathbf{c_1^n}$ denote the correctness of each step, is then used by the feedback generator to produce feedback ($\mathbf{f}$). This feedback is often limited to specifying that the answer is correct/incorrect. The biggest challenge with this framework is that it requires significant effort from the educator. We take the position that in order for these remedial classes to be effective, the problems must take into account each student's proficiency and these must be produced in large volumes in order to cater for large numbers of learners (e.g., Mathematics I at the University of Cape Town had more than 458 learners in 2011 [4]). As such, we propose to modify the architecture presented in Fig. 1 and introduce a module for automatically generating mathematics problems that abide by the South African curriculum. Such an approach would reduce effort from the educator and also allow learners to have greater control over the types of problems that they need additional practice on. For this paper, we focus only on the task of problem generation and leave out answer validation and feedback generation for future work.

To the best of our knowledge, the only information system that has the capability to generate mathematics problems geared to the South African curriculum is a proprietary product created by Siyavula[1]. Currently, it is not possible to build a system that is open to all learners because there are no published models of South African high school maths problems and the existing government-issued curriculum statements[2] are designed for humans and are not detailed so as to allow the automatic generation of problems. There is also no

[1] https://www.siyavula.com/.
[2] https://wcedeportal.co.za/eresource/106331.

controlled natural language for authoring such problems and no algorithms that can automatically generate problems. In this paper, we propose to address this gap by developing novel archetypes of high mathematics problems which we use together with a novel controlled natural language (CNL) to create the Computer Assistant for High School Mathematics (COMPAH). We have chosen to limit the scope of COMPAH to the first topic that grade 12 learners find challenging, namely *sequences and series*, after analysing the department of basic education's diagnostic reports from 2012–2020. To develop the assistant and its underlying resources, we began by creating a corpus of sequences and series problems from grade 12 exams papers from 2008–2020, analysing the corpus and formalising the problems, extracting the archetypes for the various problem types, creating templates for supported archetypes, and creating a tool that uses those resources to generate problems. We evaluated the resources by demonstrating their coverage over a held-out portion of the created corpus and showed that the identified problem types support 65% of the test set's problems. If we do limit each problem type to a specific sequence and series type, where appropriate, then coverage is 73%.

The rest of the paper is structured such that Sect. 2 examines existing work on the generation of mathematics problems, Sect. 3 introduces the problem and controlled natural language, Sect. 4 evaluates and discusses the coverage of the extracted problem types, and Sect. 6 concludes.

2 Related Work

The only information system that can generate mathematical problems for the South African curriculum is proprietary; hence, we can not determine the precise nature of how it is able to achieve this task. The company that develops the system only states that it relies on the Python programming language and custom exercise templates[3] to generate problems. As such, we have to look at the broader research community for related work. There are two strands of work that generate mathematical problems, namely, those that come from the Natural Language Processing (NLP) and educational psychology research communities.

NLP research has investigated the use of recurrent neural networks [10], answer-set programming and traditional natural language generation techniques [5], and pre-trained language models [9] to generate word problems, which may sometimes be personalised to each student. The input to such systems varies depending on the goal of their respective authors. For instance, Zhou et al.'s [10] focus is generating problems that differ in style but test the same mathematical knowledge. As such, the input to their model is a topic/style and equation—the equation is not automatically generated. The goal of Polozov et al.'s [5] work is to generate personalised maths problems and expect tutor/teacher and student requirements as input. The tutor provides the properties of the equations that underlie the problem to be generated. Specifically, they enter the operators that can be used in the equation and also provide a template for the equations.

[3] https://www.siyavulaeducation.com/technology-components.html.

The student specifies the features that drive the narrative style of the problem. For instance, they enter the theme of the story (e.g., fantasy) and the names, genders, and relationships between the story's characters. All the work on generating mathematics from the NLP community is not suitable for generating problems for South African learners, especially number patterns and sequences, since it mostly focuses on narrative generation for arithmetic and linear equation problems.

Research on the topic from the educational psychology field largely relies on templates for maths problem generation (e.g., [1–3]). The focus of such work is creating computer-adaptive tests (i.e., test items that are tailored to the learners' proficiency) and developing reliable scores for discriminating between learners of various proficiency. The major focus on such work is the latter and it uses test item theory since its models provide coefficients of test item difficulty. The only work in this area that is directly linked to the task at hand is the Adaptive Content for Evidence-based Diagnosis (ACED) prototype [6,7] that was built to support learners learning sequences and patterns at the grade 8 level in the United States (US). The system's architecture is appealing as it includes a model of the student's proficiency and the knowledge required to solve tasks, a method of updating learners' proficiency from their performance on tasks, and a model of the various tasks. However, the major limitation of the work for our task is that it is geared towards the US curriculum and its methods for problem generation are not publicly available.

Overall this means that while existing work has attractive properties from an architectural perspective, they are not sufficient to generate mathematics problems that abide by the South African mathematics curriculum.

3 COMPAH: Computer Assistant for High School Mathematics

In creating the problem archetypes, we analyse existing problems so as to create open and precise formalisation of mathematics problems.

We downloaded *sequence and series* problems from the South African Department of Basic Education's website[4] for the years 2008–2009 and 2011–2020. The year 2010 is not included because the website did not have the exam paper that was written in that year. We used the exam papers written in the November cycle to form the data to be used to formalise the problem types, build the archetypes, and create a controlled natural language. Henceforth, we shall refer to this corpus as the *development set*. We also used the papers written in the Feb/March cycle to create the *test set*. Specifically, these are the matric rewrite exam papers from 2011–2018 (inclusive).

We then analysed the problems presented in the training set in order to develop formal models of what the problem looks like. Prior to development, we first introduce some preliminaries to explain what we mean by "model".

[4] https://www.education.gov.za/.

3.1 Preliminaries

Developing models that are specific to sequence and series problems requires a general definition of what mathematical problems and solutions are. This is important because we want to distinguish between a maths problem and its description as we are not interested in only characterising problems at the linguistic-level. We take the position that a maths problem is a consistent axiomatic system (the problem setting) and a proposition (the answer). A student's solution can be defined as a proof, possibly inconsistent, of the problem's proposition. In order to demonstrate these components, consider the problem that was taken from [10]: "Joan found 70 seashells on the beach. She gave Sam some of her seashells. She has 27 seashells. How many seashells did she give to Sam?". This problem can be formalised using the axioms labelled A1-A5, the Peano axioms[5], and answer P1. The student is expected to provide a proof of proposition P1.

A1 $\forall x, y\ possesses(x, y) \rightarrow Person(x) \wedge Seashell(y)$
A2 $\forall x, y, z\ gave(x, y, z) \rightarrow Person(x) \wedge Person(y) \wedge Seashell(z)$
A3 $\forall x, y(\exists^a p\ possesses(x, p) \wedge \exists^b q\ gave(x, y, q) \wedge \exists^c r\ possesses(y, r)) \rightarrow$
$\exists^{a-b} s\ possesses(x, s) \wedge \exists^{c+b} t\ possesses(y, t)$
A4 $\exists^a s_1\ possesses(p_1, s_1) \wedge \exists^b s_1\ possesses(p_1, s_1) \rightarrow a = b$
A5 $\exists^a y\ possesses(sam, y) \wedge \exists^{70} y\ possesses(joan, y) \wedge \exists^x z\ gave(joan, sam, z) \rightarrow$
$\exists^{27} y\ possesses(joan, y) \wedge \exists^{a+x} y\ possesses(sam, y)$
P1 $x = 43$

Due to space limitations, we do not include a proof of the above problem. Using the above intuitive view as a basic, we present Definitions 1 and 2 in order to be precise on what problems and solutions are.

Definition 1 (Maths problem). *Let MP denote the set of maths problem and A denote the set of answers to a maths problem. We define a maths problem as the quadtruple $\langle AS, \alpha, p, q\rangle \in MP$ where the following conditions are met:*

- *$AS \subseteq S_{FOL}$ is a finite and consistent set of First Order Logic sentences*
- *$AS = DA \cup PA$ where DA are domain/theory axioms and PA are axioms of the problem*
- *$\alpha_1, ..., \alpha_n \in A \subseteq S_{FOL}$ are tautologies. They are the solutions to the problem*
- *p is a natural language description of the premise of the problem*
- *q is a natural language description of the question of the problem*

Definition 2 (Maths solution). *Let SO denote the set of solutions for a maths problem. We define a solution for some maths problem $p \in MP$ as the finite ordered list $\langle s_1, s_2, ..., s_m\rangle \in SO$ where $s_m = \alpha_p$ and $\forall 1 \leq j \leq m\ s_j$ one of the following conditions hold:*

[5] Omitted here for brevity.

– α_p *is a solution to the problem*
– $s_j \in AS_p$ *where* AS_p *is the problem's set of FOL sentences*
– s_j *is a tautology*
– $\exists\ 1 \leq g, h < j$ *such that* $s_g \wedge s_h \rightarrow s_j$ *can either be true or false*

Given the above definitions, we define a problem archetype as a template of a maths problem. Archetypes are formed by taking a problem $\langle AS, \alpha, p, q\rangle$ and its solutions A and modifying the sentences in AS and A such that there are statements have place-holders that can take different values. That allows one to create different instances of the same problem type. With these definitions in hand, we now turn to how we categorise sequences and series' in order to have a manageable organisation of the formalised problem types.

3.2 Categorisation of Problems

The development set initially had a total of 111 sequences and series problems but we filtered out 23 problems. The 23 problems were filtered out because they were either multimodal (i.e., text and images), the premise used a very specialised narrative that would necessitate the creation of a lexico-syntactic CNL[6], the sequence or series type could not be determined, or the question is judged to be unlikely to be repeated in the future.

Analysis of the remaining 88 sequences and series problems in the training set showed that there are four types of sequences, and series based on them, that are examined in the papers and we present the first three in Definition 3.

Definition 3. *Let G and A denote geometric and arithmetic geometric sequences, respectively. We define each of the possible sequence types as ordered sets in the following manner:*

– $G = \{x \mid x = (\{x_1\} \cup \bigcup_{i=2}^{\infty} \{x_i = f(i)\}, O_i)\}$ *where* $f(i) = x_1 r^{i-1}$
– $A = \{x \mid x = (\{x_1\} \cup \bigcup_{i=2}^{\infty} \{x_i = f(i)\}, O_i)\}$ *where* $f(i) = x_1 + (i-1)d$ *and* $d \in \mathcal{N}$ *is a constant*
– $Q = \{x \mid x = (\bigcup_{i=1}^{\infty} \{f(i)\}, O_i)\}$ *where* $f(x) = ax_1^2 + bx_1 + c$

The above sequence types have also been used in exams papers to form a sequence type that possesses an interleaving constant. For instance, the quadratic sequence (2, 3, 10, 23, ...) and the constant 0 can be used to form the new sequence type (0, 2, 0, 3, 0, 10, 0, 23, ...). We call this an interleaving sequence and define it in the following manner:

Definition 4. *Let I denote the interleaving sequence type. We define interleaving sequences in the following manner:*

[6] A design decision was taken to not have lexico-syntactic patterns in the current version to reduce complexity.

– $S = A \cup G \cup Q$
– $CI = \{y \mid y = \{(1-\alpha^i)k + \alpha^i x_i\}, O_i)\}$ *where* $i \in [1, \infty)$, $\alpha^i = i \mod 2$, $k \in \mathcal{N}$, *and* $x_i \in X \subseteq S$
– $SI = \{y \mid y = \{(1-\alpha^i)x_i + \alpha^i y_i\}, O_i)\}$ *where* $i \in [1, \infty)$, $\alpha^i = i \mod 2, y_i \in Y \subseteq S, x_i \in X \subseteq S$
– $I = CI \cup SI$

Using the sequence and series types we then analysed and formalised problems along these categories. We then formalised the remaining 88 problems by identifying the axioms that make up the problem setting and their answers, following from Definition 1.

3.3 Problem Types

After examining the axioms and solutions of the 88 formalised problems in the *development set* that use the various sequences and series types, we found a total of 64 archetypes or problem types. We present them here using a mixture of natural language and mathematical notation for ease of reading. We decided against only listing the formal problem (i.e., as quadruples of the form $\langle AS, \alpha, p, q\rangle$ following from Definition 1) for the benefit of human readers.

Archetypes for arithmetic sequences and series

1. Given $s_1 = f(x), s_2 = g(x), s_3 = h(x)$ for some $s \in A$. Show that $s_n = f(x) + (n-1)(h(x) - g(x))$
2. Given $s_1 = f(x), s_2 = g(x), s_3 = h(x)$ for some $s \in A$. Show that $s_i = \alpha$ for $i > 3$
3. Given $s_1 = f(x), s_2 = g(x), s_3 = h(x)$ for some $s \in A$. Show that $\sum_{i=1}^{n} s_i = \alpha$ for $n > 4$
4. Given $s_1 = \alpha, s_2 = \beta$ for some $s \in A$. Show that $s_i = \eta$ for some $i > 2$
5. Given $s_1 = \alpha, s_2 = \beta$ for some $s \in A$. If $\sum_{i=1}^{n} s_i = \pi$, show that $n = \zeta$.
6. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, ..., s_i = \eta$ for some $s \in A$. Show that $s_n = \alpha + (n-1)(s_{i+1} - s_i)$ for some i.
7. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, ..., s_i = \eta$ for some $s \in A$. If $s_i \equiv x \mod a$ for $i \in [start, end]$, show that $x \in \{x_1, ..., x_n\}$
8. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, ..., s_i = \eta$ for some $s \in A$. If $s_i \equiv x \mod a$, show that $\sum_{i=1}^{n} s_i = \pi$.
9. Given $s_1 + s_2 + s_3 + ... + s_k$ for some $s_n \in A$. Express the series using sigma notation.
10. Given $s_1 + s_2 + s_3 + ... + s_k$ for some $s_n \in A$. Show that $l = |\{s_i \mid s_i \equiv \omega \mod \lambda\}|$
11. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, ..., s_{i-1} = \delta, s_i = \epsilon, ..., s_j = \zeta$ for some $s_n \in A$. Show that $k = |\{s_i \mid s_i \equiv \pi \mod \lambda\}|$
12. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, ..., s_{i-1} = \delta, s_i = \epsilon$ for some $s_n \in A$. Show that $i = \eta$.

13. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, ..., s_{i-1} = \delta, s_i = \epsilon$ for some $s_n \in A$. Show that $\sum_{k=1}^{i} s_k = \eta$ for $s_k < 0$
14. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$ for some $s_n \in A$. If $\sum_{i=1}^{n} s_i = \pi$, show that $n = \zeta$
15. Given $s_1 = \alpha, s_4 = \beta$ for some $s_n \in A$. Show that $s_2 = \zeta$ and $s_3 = \eta$.
16. Given $s_n = f(n)$ for some $s_n \in A$. Show that $s_i = \alpha, s_{i+1} = \beta$, and $s_{i+2} = \gamma$.
17. Given $s_n = f(n)$ for some $s_n \in A$. Show that $\sum_{i=1}^{n} f(i) = \alpha$
18. Given $s_1 = \alpha, s_2 = f(x)$, and $s_3 = \beta$ for some $s_n \in A$. Show that $x = \gamma$.
19. Given $s_1 = \alpha$ and $d = s_{i+1} - s_i, \forall i \in \mathcal{Z}^+$. Show that $s_n = \frac{n}{2}(2\alpha + (n-1)d)$
20. Given $s_i = f(x), s_{i+1} = g(x)$, and $s_{i+2} = h(x)$ for some $s_n \in A$. Show that $x = \alpha$.
21. Given $s_1 + s_2 + s_3 + ... + s_k = \pi$ for some $s_n \in A$. Show that $s_i = \mu$ where $3 < i < k$.
22. Given $s_1 + s_2 + s_3 + ... + s_k$ for some $s_n \in A$. Show that $\sum_{i=1}^{k} = \mu$

Archetypes for geometric sequences and series

1. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$ for some $s_n \in G$. If $s_n = \alpha r^{n-1}$, show that $r = \zeta$
2. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$ for some $s_n \in G$. Does $\sum_{i=1}^{\infty} s_i$ converge?
3. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$ for some $s_n \in G$. Specify reasons why $\sum_{i=1}^{\infty} s_i$ converges?
4. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$ for some $s_n \in G$. Show that $\sum_{i=1}^{\infty} s_i = \zeta$
5. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$ for some $s_n \in G$. Show that $\sum_{i=1}^{n} s_i = \zeta$
6. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$ for some $s_n \in G$. Show that $\sum_{i=1}^{\infty} s_i - \sum_{i=1}^{n} s_i = ab^n$ for some a, b.
7. Given $s_1 = \alpha, s_2 = ks_1, ..., s_j = ks_{i-1}, ...$ for some $s_n \in G$. Show that the proposition $s_1 = \sum_{i=2}^{j} s_i$ is true/false.
8. $s_1 = \alpha, s_3 = f(x), s_3 = \beta$ for some $s_n \in G$. Show that $x = \eta$.
9. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$ for some $s_n \in G$. Show that $s_i = \eta$ for some $i > 3$
10. Given $\sum_{i=1}^{n} f(x,i)$ for some $f(x,i) \in G$. Show that $\sum_{i=1}^{n} f(x,i) = \zeta$ if $x = \alpha$.
11. Given $\sum_{i=1}^{n} f(x,i)$ for some $f(x,i) \in G$. If $\sum_{i=1}^{\infty} f(x,i)$ converges, show that $x \in \{x_1, ..., x_n\}$.
12. Given $s_1 = \alpha, r = \beta, \sum_{i=3}^{\infty} s_i = \gamma$ for some $s_i \in G$. If $s_i + s_{i+1} = \delta$, show that $s_i + s_{i+1} = f(a, r)$

13. If $\sum_{i=1}^{k} \alpha_1 \beta_1^{f(k)} = p$, Show that $\sum_{i=1}^{k} \alpha_2 \beta_2^{g(k)} = f(p)$
14. Given $s_n = f(n)$ for some $s_n \in G$. Show that $s_i = \alpha$.
15. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$, and $s_4 = \delta$ for some $s_n \in G$. Show that $s_i = \pi$ for $i > 4$.
16. Given $r = \frac{s_{i+1}}{s_i}$ and $\sum_{i=1}^{\infty} s_1 r^{i-1} = \alpha$. Show that $s_i = \alpha$.
17. Given $s_n = f(n)$ for some $s_n \in G$. If $\sum_{i=1}^{n} s_i = \alpha$, show that $n = \beta$.

Archetypes for quadratic sequences and series

1. Given $s_n = f(n)$. Show that $s_i = \alpha, s_{i+1} = \beta, s_{i+2} = \gamma$
2. Given $s_n = f(n)$. If $s_j = min(f(n))$, show that $j = \alpha$.
3. Given $s_n = f(n)$. If $d_i = s_i - s_{1+1}$ and $d_{i+1} = s_{i+2} - s_{i+3}$, show that $d_1 - d_2 = \alpha$.
4. Given $s_n = f(n)$. If $s_i < \alpha$, show that $i \in i_1, ..., i_n$
5. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$. Show that $s_i = \eta$ for some $i > 3$.
6. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$. Show that $s_n = an^2 + bn + c$ for some a, b, c
7. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma$. If $s_{i+1} - s_i = \psi$ for some i, show that $s_{i+1} = \eta$ and $s_i = \zeta$.
8. Given $s_1 + s_2 + s_4 + ...$ where $s_n \in Q$. Show that $s_n = an^2 + bn + c$ for some a, b, c
9. Given $s_1 + s_2 + s_4 + ...$ where $s_n \in Q$. Express the series using sigma notation.
10. Given $s_1 + s_2 + s_4 + ...$ where $s_n \in Q$. Show that the proposition $\sum_{i=1}^{n} s_i = f(n)$ is true/false.
11. Given $s_1 = \alpha, s_2 = g(x), s_3 = \beta, s_4 = f(x)$ for some $s_n \in Q$. Show that $x = \zeta$.
12. Given $s_1 = \alpha, s_2 = g(x), s_3 = \beta, s_4 = f(x)$ for some $s_n \in Q$. If $d_i = s_{i+1} - s_i$ and $\sum_{j=1}^{n} d_j > \mu$, show that $n \in \{n_1, ..., n_m\}$
13. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, s_4 = \delta$ for $s_n \in Q$. If $d_i^1 = s_{i+1} - s_1$ and $d_i^2 = d_{i+1}^1 - d_i^1$, show that $d_1^2 = \zeta$
14. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, s_4 = \delta$ for $s_n \in Q$. If $s_i = \eta$, show that $i = \zeta$.
15. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, s_4 = \delta$. If $d_i = s_{i+1} - s_i$, show that $\sum_{i=1}^{n} d_i = f(n)$ is true/false.
16. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, s_4 = \delta$. If $\sum_{i=1}^{n} d_i = \eta$, show that $n \in \{n_1, ..., n_m\}$
17. Given $\sum_{i=p}^{\infty} \alpha \beta^{f(i)} = k$. Show that $p = \eta$.
18. Given $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, s_4 = \delta$, and $s_5 = \epsilon$ for some $s_n \in Q$. If $d_i = s_{i+1} - s_i$, show that $d_n = f(n)$.
19. Given $s_i = \alpha, s_{i+1} = \beta$, and $s_{i+3} = \gamma$. If $d_i^1 = s_{i+1} - s_i$ and $d_i^2 = d_{i+1}^1 - d_i^1$, show that $d_i^2 = \eta, \forall i \in \mathcal{Z}^+$
20. Given $s_i = \alpha, s_{i+1} = \beta$, and $s_{i+2} = \gamma$ for some $s_n \in Q$. Show that $s_j = \eta$.
21. Given $s_1 = \alpha, s_2 = f(x), s_3 = \beta$, and $s_4 = h(x)$. Show that $x = \mu$.

Archetypes for interleaved sequences

1. Given $s_1 = \alpha, s_2 = \beta, ..., s_{11} = \lambda$ for some $s_n \in I$. Show that $s_i = eta$ for some $i > 11$.
2. Given $s_1 = \alpha, s_2 = \beta, ..., s_{11} = \lambda$ for some $s_n \in I$. Show that $\sum_{i=1}^{n} s_i = \eta$ for some $n > 11$.
3. $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, s_4 = \delta, s_5 = \epsilon$, and $s_6 = \zeta$ for some $s_n \in I$. Show that $s_i - sj = \eta$ for some $i, j \in \mathcal{Z}^+$.
4. $s_1 = \alpha, s_2 = \beta, s_3 = \gamma, s_4 = \delta, s_5 = \epsilon$, and $s_6 = \zeta$ for some $s_n \in I$. Show that $s_i \equiv a(mod\ b), \forall i \in \mathcal{Z}^+$.

The mathematical formalisations are useful for being precise on the structure of mathematics problems. We now turn to the evaluation of the scope on the test set.

4 Evaluation of Scope

Analysis of the 74 questions found in the test set showed that 48 (65%) fell within the scope of the problem types extracted from the development set. Six of the other 26 questions were not directly covered by the extracted problem types but they were testing knowledge that is covered by the extracted problem types. For instance, question 3.2.1 from 2011 asked learners to calculate the sum of the first 20 items of geometric series and it was not captured by the extracted problem types, however, the same kind of question was asked in the building set in the context of an arithmetic series. This means that the external problem types cover 73% of the problems in the test set if we do not limit each problem type to a specific sequence or series type.

5 Utility of Archetypes

To demonstrate utility, we extracted templates to construct a CNL and used Python and the symbolic mathematics package SymPy[7] to capture the archetypes for the first fifteen arithmetic sequence and series problem types and developed a prototype assistant that makes use of the archetypes and templates to generate text. The prototype currently uses the architecture shown in Fig. 2. In the rest of section, we discuss these components and how they work together to generate a natural language problem description.

The problem generation module in Fig. 2 takes an identifier of a problem type, selects and populates the place-holders found in the archetype axioms using random values, and produces an instance of an abstract representation of the problem. For instance, when the module is given the identifier 4, it retrieves the archetype shown in Listing 1.1. The module then generates random values for $s_1, s_2 3$, and i the place-holders found in line 2 of Listing 1.1. Those values

[7] https://www.sympy.org/en/index.html.

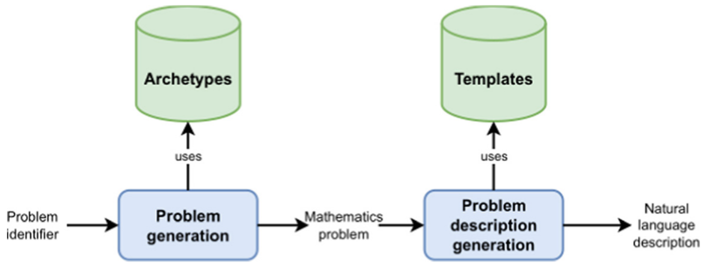

Fig. 2. Architecture used by prototype assistant

are then used to create concrete values for the premises of the problem (lines 11–13) and the solution (line 16). The resulting mathematics problem is then fed to the question generation module.

```
 1  class ArithmeticArchetypeFour(Archetype):
 2      def __init__(self, s1, s2, i):
 3          super().__init__()
 4          s1_symbol = Symbol('s1')
 5          s2_symbol = Symbol('s2')
 6          i_symbol = Symbol('i')
 7
 8          prem1 = Eq(s1_symbol, s1)
 9          prem2 = Eq(s2_symbol, s2)
10          prem3 = Eq(i_symbol, i)
11          self.premises.append(prem1)
12          self.premises.append(prem2)
13          self.premises.append(prem3)
14
15          self.seq = ArithmeticSequence(s1, s2 - s1, 's')
16          self.solution = self.seq.get_function()(i)
```

Listing 3.1. Archetype that can be used to generate arithmetic sequence problems of type 4

The problem description module takes in the maths problem, fetches the associates templates from the CNL, and then inserts the slot values in the templates using the information found in the Python archetype. For instance, for the problem created using the archetype in in Listing 1.1, the module can select the templates `Consider the sequence: [sequence]` and `If the pattern continues in the same manner, determine [seqItemLabel]` for the premise and question respectively. In the premise template, the slot `[sequence]` is a place-holder for s_1 and s_2. In the question template, the slot `[seqItemLabel]` is a place-holder for the label for the ith element. The module can then used those premise and question templates to generate problems of the form:

```
Consider the sequence: 5, 17
If the pattern continues in the same manner, determine s62.
```

The python archetypes, controlled natural language, and prototype implementation are released as supplementary material at https://zenodo.org/record/6927550.

6 Conclusion

This paper presented the first formalisation of mathematical problems that high school leaving learners are expected to master. This formalisation is based on the analysis of problems from a corpus of problems covering approximately 12 years of South African grade 12 exams. We evaluated the coverage of our formalisations on problems taken from the Matric rewrite exam papers from 2011–2018 (inclusive). We have found that the formalised problems cover 65% of the problems in the test set. Moreover, when we do not limit each problem formalisation to a specific sequence and series type, where appropriate, then we see that our formalised problem types cover 73% of the problems. We have also built a prototype assistant that relies on archetypes based on the formalised problem types and a controlled natural language to demonstrate the utility of the formalisations. The created artefacts can be used by lecturers to enable student-led revision of high school mathematics thus improving students' mathematics background. Other researchers may find the artefacts useful in investigating the diversity of maths problems over the years, how differences in the curriculum impact the actual problem posed to students, etc.

As current and future work, we focus on extending the archetypes and the controlled natural language to cover other problem types, introducing models of student proficiency, and investigating the impact of the assistant on educational outcomes.

References

1. Deane, P., Graf, E.A., Higgins, D., Futagi, Y., Lawless, R.: Model analysis and model creation: the task-model structure of quantitative item domains. ETS Research Report Series 2006, no. 1, pp. i–63 (2006)
2. Embretson, S.E., Kingston, N.M.: Automatic item generation: a more efficient process for developing mathematics achievement items? J. Educ. Measur. **55**(1), 112–131 (2018)
3. Holling, H., Bertling, J.P., Zeuch, N.: Automatic item generation of probability word problems. Stud. Educ. Eval. **35**(2), 71–76 (2009)
4. Jawitz, J.: The Challenge of Teaching Large Classes in Higher Education in South Africa?: A Battle to be Waged Outside the Classroom, vol. 4. African Sun Media, Cape Town (2013)

5. Polozov, O., O'Rourke, E., Smith, A.M., Zettlemoyer, L., Gulwani, S., Popovic, Z.: Personalized mathematical word problem generation. In: Yang, Q., Wooldridge, M.J. (eds.) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, 25–31 July 2015, pp. 381–388. AAAI Press (2015)
6. Shute, V.J., Graf, E.A., Hansen, E.G.: Designing adaptive, diagnostic math assessments for individuals with and without visual disabilities. ETS Research Report Series 2006, no. 1, pp. i–37 (2006)
7. Shute, V.J., Hansen, E.G., Almond, R.G.: An assessment for learning system called aced: Designing for learning effectiveness and accessibility. ETS Research Report Series 2007, no. 2, pp. i–45 (2007)
8. Subban, A.: Guidelines for the development of the school timetables - reopening for schools Covid-19. Technical report, Department of Basic Education (2020). https://wcedeportal.co.za/eresource/131181
9. Wang, Z., Lan, A.S., Baraniuk, R.G.: Math word problem generation with mathematical consistency and problem context constraints. In: Moens, M., Huang, X., Specia, L., Yih, S.W. (eds.) Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event/Punta Cana, Dominican Republic, 7–11 November 2021, pp. 5986–5999. Association for Computational Linguistics (2021)
10. Zhou, Q., Huang, D.: Towards generating math word problems from equations and topics. In: van Deemter, K., Lin, C., Takamura, H. (eds.) Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019, Tokyo, Japan, 29 October–1 November 2019, pp. 494–503. Association for Computational Linguistics (2019)