AI Communications 31 (2018) 251–266
DOI 10.3233/AIC-180762
IOS Press

Can an A.I. win a medal in the mathematical olympiad? – Benchmarking mechanized mathematics on pre-university problems[1]

Takuya Matsuzaki [a,*], Hidenao Iwane [b], Munehiro Kobayashi [c], Yiyang Zhan [d], Ryoya Fukasaku [e], Jumma Kudo [e], Hirokazu Anai [b], and Noriko H. Arai [f]
[a] *Nagoya University, Japan*
E-mail: matuzaki@nuee.nagoya-u.ac.jp
[b] *Fujitsu Laboratories, Ltd., Japan*
E-mails: iwane@jp.fujitsu.com, anai@jp.fujitsu.com
[c] *University of Tsukuba, Japan*
E-mail: munehiro-k@math.tsukuba.ac.jp
[d] *Université Paris Diderot, France*
E-mail: pon.zhan@gmail.com
[e] *Tokyo University of Science, Japan*
E-mails: ryoya.0323@gmail.com, 1414606@alumni.tus.ac.jp
[f] *National Institute of Informatics, Japan*
E-mail: arai@nii.ac.jp

Abstract. This paper introduces a benchmark problem library for mechanized mathematics including computer algebra and automated theorem proving. The library consists of pre-university mathematical problems taken from exercise problem books, university entrance examinations, and the International Mathematical Olympiads. The subject areas include real algebra, geometry, number theory, pre-calculus, calculus, and combinatorics. It thus includes problems in various areas of pre-university mathematics and with a variety of difficulty. Unlike other existing benchmark libraries, this one contains problems that are formalized so that they are obtainable as the result of mechanical translation of the original problems expressed in natural language. In other words, the library is designed to support the integration of the technologies of mechanized mathematics and natural language processing towards the goal of end-to-end automatic mathematical problem solving. The paper also presents preliminary experimental results of our prototype reasoning component of an end-to-end system on the library. The library is publicly available through the Internet.

Keywords: Mechanized mathematics, benchmark library

1. Introduction

One of the ultimate goals of automated theorem proving is to produce computer programs that allow a machine to conduct mathematical reasoning like human beings. It seems that a tacit understanding exists on how we should interpret this goal. First, the input of the programs is assumed to be expressed in some formal language, but not in a natural language. Second, the term "human beings" is used to mean gifted mathematicians rather than ordinary people. In this paper, we propose a different interpretation of the goal by providing a new problem library for benchmarking automated mathematical reasoners, and showing experimental results on the problem set.

[1]This is an extended and updated version of a talk at the 1st Conference on Artificial Intelligence and Theorem Proving and a paper entitled "Race against the Teens – Benchmarking Mechanized Math on Pre-university Problems" published in the proceedings of the International Joint Conference on Automated Reasoning 2016 (IJCAR 2016) [33]. All the statistics and the experimental results are updated using the latest public version of the benchmark data and the current version of the prototype solver.

[*]Corresponding author. E-mail: matuzaki@nuee.nagoya-u.ac.jp.

We developed a new problem library of real pre-university mathematical problems. It is designed to cover various sub-areas of curriculum mathematics and a diverse range of difficulty. The initial release of the data set includes more than 700 problems taken from three sources: popular high school exercise book series, entrance examinations of seven top universities in Japan, and the past problems from International Mathematical Olympiads (IMO). Our choice of the three problem sources is motivated by the desire to measure the performance of mechanized mathematics systems with high school students of different skill and intellectual levels as reference points.

Although problems in the library are formalized in a formal language so that the automatic reasoning (AR) and computer algebra communities will find it appealing to challenge the problems, the formalization is designed so that the problems can be obtained as the result of mechanical translation of their originals. This might have sounded unrealistic in the previous century but is now within the range of contemporary research, thanks to the recent progress that has been made in deep linguistic processing (e.g., [8,12,30]).

Figure 1 presents an output from the translation module under development. The input is a problem taken from IMO:

> The diagonals AC and CE of the regular hexagon $ABCDEF$ are divided by the inner points M and N, respectively, so that $\frac{AM}{AC} = \frac{CN}{CE} = r$. Determine r if B, M, and N are collinear.
>
> IMO 1982, Problem 5

Roughly speaking, the first half of the formula in Fig. 1 (i.e., `(exists(F)...)`) says $ABCDEF$ is a regular hexagon and AC and CE are its diagonals. The second polygon $PQRSTU$ is introduced by the definition of the term "regular hexagon", which states that a polygon x is a regular hexagon if and only if there exists a regular polygon with six vertices $(P, Q, \dots, U)$ that is equal to x.

The second half (i.e., `(exists(M N)...)`) says the points M and N satisfy $\frac{AM}{AC} = \frac{CN}{CE} = r$, they are inner points of AC and CE, and B, M, and N are collinear.

Figure 2 depicts a part of the process that derives the logical translation of a Japanese phrase "正6角形/regular-hexagon $ABCDEF$ の/of 対角線/diagonal", that corresponds to "diagonal(s) of the regular hexagon $ABCDEF$." The mechanical translation is based on a grammar formalism called combinatory categorial grammar (CCG) [42]. The translation process starts by fetching the *lexical items* defined in a dictionary. A lexical entry is a triple of a word, its syntactic category, and its semantic function, e.g.:

$$\mathit{diagonal} :: N_{sg,2d.Shape}/NP_{of,sg,2d.Shape}$$
$$: \lambda y \lambda x\big(\texttt{is-diagonal-of}(x, y)\big).$$

The lexical entries correspond to the leaves of the derivation tree. The semantic functions are combined into the semantic representation of a sentence according to the constraints encoded in the syntactic categories. For instance, the forward application rule (>) combines the semantic functions of two words (or phrases) having syntactic categories of the form X/Y and Y as follows:

$$> \frac{X/Y : f \quad Y : y}{X : f(y)}$$

Currently, the dictionary contains 55,000 lexical items for over 8,000 word forms. By manually inspecting the output of the translation module, we verified that it derives a semantic representation on 70% of unseen sentences with the accuracy of 90% (see [31] for further details).

In parallel with the development of the dictionary and the translation module, our problem library was developed by *manually* formalizing the mathematical problems. It has been used as a substitute for the output of the translation module in the development of the reasoning module of the end-to-end system. Problems in the library were thus manually formalized according to the design of the translation module. That is, they were translated manually into the formal language on a word-by-word and sentence-by-sentence basis without any inference and paraphrasing.

The formalized problem set and the accompanying axioms are publicly available.[2] The problems and the axioms are formulated in a higher-order language that is compatible with the typed higher-order form with rank-1 polymorphism (TH1) [25] of 'Thousands of Problems for Theorem Provers' (TPTP) [44]. The data and the axioms are distributed both in TPTP's TH1 syntax and Lisp S-expression format. For readability, we use the S-expression format for presenting the data. Several basic elements such as logical connectives and quantifiers are renamed following TPTP's convention.

The development of the data set and a prototype solver system has been carried out as a part of a long-term AI research project called "Todai Robot Project".

[2] The URL is: https://zenodo.org/record/815138#.WUojn8bnpfh.

```
(Find (x)
  (exists (A B C D E)
     (& (exists (F)
           (& (exists (U T S R Q P)
                 (& (= (polygon (list-of A B C D E F))
                       (polygon (list-of P Q R S T U)))
                    (is-regular (polygon (list-of P Q R S T U)))))
              (is-diagonal-of (seg A C) (polygon (list-of A B C D E F)))
              (exists (U T S R Q P)
                 (& (= (polygon (list-of A B C D E F))
                       (polygon (list-of P Q R S T U)))
                    (is-regular (polygon (list-of P Q R S T U)))))
              (is-diagonal-of (seg C E) (polygon (list-of A B C D E F)))))
        (exists (M N)
           (& (exists (r)
                 (& (= (/ (length-of (seg A M)) (length-of (seg A C)))
                       (/ (length-of (seg C N)) (length-of (seg C E))))
                    (= (/ (length-of (seg C N)) (length-of (seg C E))) r)
                    (on M (seg A C))
                    (= (/ (length-of (seg A M)) (length-of (seg A C)))
                       (/ (length-of (seg C N)) (length-of (seg C E))))
                    (= (/ (length-of (seg C N)) (length-of (seg C E))) r)
                    (on N (seg C E))
                    (= x r)))
              (~ (= M A)) (~ (= M C))
              (~ (= N C)) (~ (= N E))
              (points-colinear (list-of B M N)))))))
```

Fig. 1. Mechanical translation result of IMO 1982, Problem 5.

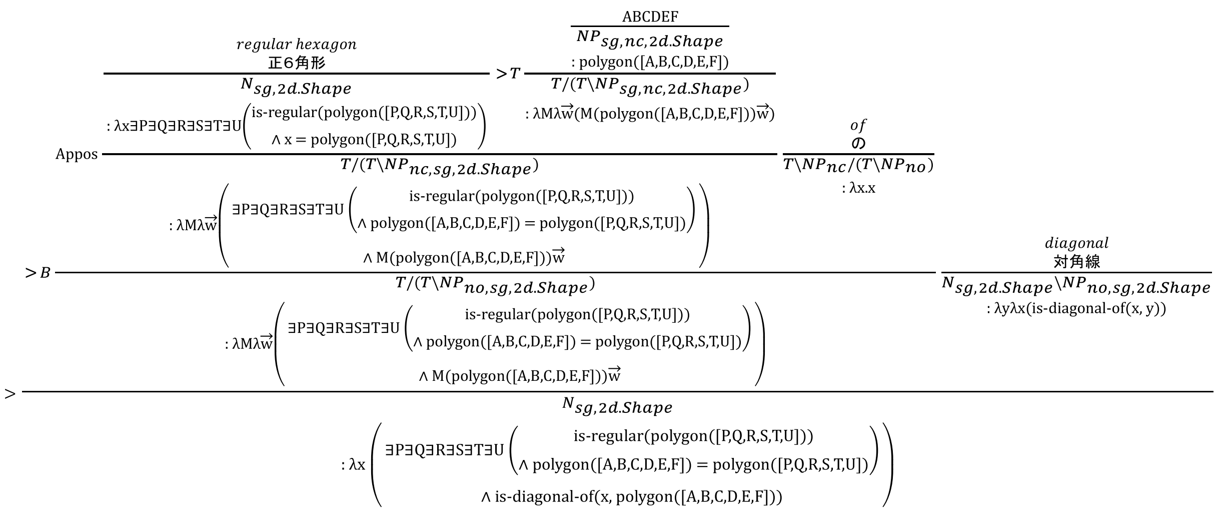
regular hexagon
正6角形
$N_{sg,2d.Shape}$
$: \lambda x \exists P \exists Q \exists R \exists S \exists T \exists U \left(\begin{array}{c} \text{is-regular(polygon([P,Q,R,S,T,U]))} \\ \wedge\, x = \text{polygon([P,Q,R,S,T,U])} \end{array} \right)$
ABCDEF
$NP_{sg,nc,2d.Shape}$
: polygon([A,B,C,D,E,F])
$>T$
$T/(T\backslash NP_{sg,nc,2d.Shape})$
$: \lambda M \lambda \vec{w}(M(\text{polygon([A,B,C,D,E,F])})\vec{w})$
Appos
$T/(T\backslash NP_{nc,sg,2d.Shape})$
of
の
$T\backslash NP_{nc}/(T\backslash NP_{no})$
$: \lambda x.x$
$>B$
$T/(T\backslash NP_{no,sg,2d.Shape})$
diagonal
対角線
$N_{sg,2d.Shape}\backslash NP_{no,sg,2d.Shape}$
$: \lambda y \lambda x(\text{is-diagonal-of}(x, y))$
$>$
$N_{sg,2d.Shape}$
$: \lambda x \left(\exists P \exists Q \exists R \exists S \exists T \exists U \left(\begin{array}{c} \text{is-regular(polygon([P,Q,R,S,T,U]))} \\ \wedge\, \text{polygon([A,B,C,D,E,F])} = \text{polygon([P,Q,R,S,T,U])} \end{array} \right) \wedge\, \text{is-diagonal-of(x, polygon([A,B,C,D,E,F]))} \right)$

Fig. 2. Syntactic/semantic analysis of problem (CCG derivation tree).

The project aims at developing AI systems that are capable of solving real university entrance exam problems including those in mathematics and other subjects such as English, physics, and world history. The final goal of the project is to 'pass' the entrance exam of the University of Tokyo (a.k.a. "Todai" in Japan) by 2021.[3] The objective of the project is to re-integrate the achievements in AI and related areas that have emerged after the fragmentation of the field in the 1980s and 1990s.

The rest of the paper is structured as follows. We first describe how we collected and formalized curriculum mathematics problems in Section 2. Several problems are shown in Section 3 to exemplify what aspects of mechanized mathematics are necessary. Besides proof problems, the benchmark includes many "Find X"-type problems. Technical issues in formalizing such problems are discussed in Section 4. Section 5 provides an overview of a prototype solver sys-

[3] http://spectrum.ieee.org/robotics/artificial-intelligence/can-an-ai-get-into-the-university-of-tokyo

tem built on an integration of a simple logical inference system with computer algebra systems. Experimental results on the initial release of the data by our solver system are presented in Section 6. Finally, we conclude the paper and discuss future directions in Section 7.

2. Pre-university mathematical problems as a benchmark for mechanized mathematics systems

In this section, we first describe the sources and the types of the benchmark problems. We then explain how we encoded the problems other than proof problems. Finally, the representation language is described.

2.1. *The problem library*

The latest release of the data set (version 1.1) consists of 743 problem files containing 1,337 directives, and 2,787 axioms defining 1,372 symbols (functions, predicates, and constants). The problems were taken from three sources: exercise books (**Ex**), Japanese university entrance exams (**Univ**), and International Mathematical Olympiads (**IMO**).

The **Ex** division of the data is taken from a popular problem book series, "Chart-Shiki" [41], which contains more than ten thousand problems in total. In the first release of the data set, the **Ex** division consists of arithmetic problems and various types of geometry problems (including those involving calculus and linear algebra) (Table 1). Every problem in the book series is marked with one to five stars by the editors of the book series according to its difficulty. We sampled the problems so that their levels of difficulty and the topics of the problems would be uniformly distributed.

The **Univ** division of the data consists of the past entrance examinations of seven top Japanese national universities.[4] Unlike in most countries, in Japan each national university prepares its entrance exam by itself. As a result, several hundreds of brand-new problems are produced every year for the entrance exams. In the first release, the **Univ** division includes the problems that were manually classified as 'most likely expressible' in the first-order theory of real-closed fields (RCF) (Table 1). We exhaustively selected such problems in the examinations held in odd numbered years from 1999 to 2013. Two hundred more **Univ** problems involving transcendental functions and integer arithmetic (often as a mixture with reals) are currently under preparation for the second release of the data set.

[4]Hokkaido University, Tohoku University, the University of Tokyo, Nagoya University, Osaka University, Kyoto University, and Kyushu University.

Table 1
Subject areas (Ex & Univ)

	Ex	Univ
Algebra	51	9
Linear Algebra	28	62
Geometry	136	65
Pre-Calculus	15	75
Calculus	42	32
Combinatorics	16	0
Total	288	243

Table 2
Subject areas (IMO)

	IMO
Algebra	57
Number Theory	38
Analysis	1
Geometry	105
Combinatorics	11
Total	212

The **IMO** division consists of about 2/3 of the past IMO problems. The initial release includes all of the geometry and real algebra problems in the IMOs held between 1959 and 2014, and some of the problems in number theory, function equations, and combinatorics.

Each problem is labeled by its subject domain name such as geometry or calculus (Table 1 and Table 2), and also by its formal theory name. The problems that are naturally expressible (by humans) in the theories of RCF or Peano Arithmetic (PA) are labeled so, and the rest of the problems are tentatively labeled ZF, standing for Zermelo-Fraenkel Set Theory. We scrutinized the problems labeled ZF and classified them into several groups such as 'RCF + PA' (mixture of integer and real arithmetics) and 'Transc' (problems involving transcendental functions that cannot be reformulated in RCF), though they are not formal theories (Table 3).

Table 4 lists statistics for the formalized problems. For reference, it also lists those for the typed higher-order format (THF) problems in TPTP version 6.4.0. The statistics about the TPTP-THF problems include the numbers of formulas, atoms, etc., in the conjectures to be proved as well as those in the axioms. The statistics about our problem library include only those numbers of the directives but *not* those of the axioms. As

Table 3

Distribution of theory labels

	Ex	Univ	IMO
PA	84	0	42
RCF	174	243	116
ZF	30	0	54
RCF + PA	1	0	8
Transc	23	0	5
PA + Transc	5	0	1
Comb	0	0	10
Other	1	0	30

Table 5

Number of problems and directives

		Ex	Univ	IMO	Total
#Problems		288	243	212	743
#Directives	`Find`	472	426	108	1006
	`Draw`	28	24	0	52
	`Show`	78	66	135	279

can be seen in the table, the average numbers of atoms, symbols, lambda abstractions (λ), and existential quantifiers ($\exists$) per formula are much larger in our dataset than in TPTP-THF problems. The number of universal quantifiers ($\forall$) is smaller because we don't count those in the axioms. We may hence say the problem statements (directives) in our dataset tend to be more complex than the conjecture formulas in TPTP-THF.

2.2. *A formalization of curriculum mathematics problems*

We formalize a problem as a pair of a *directive* and its *answer*. By surveying the problems, we identified three major types of directives:

- `Show`$[\phi]$ is a proof problem to prove ϕ.
- `Find`$(v)[\phi(v)]$ is a problem to find all values for v that satisfy condition $\phi(v)$.
- `Draw`$(v)[\phi(v)]$ requests a geometric object v defined by $\phi(v)$ be drawn.

`Show` directives must be familiar to the reader, though the set of problems requiring proofs is a minority in curriculum mathematics. Table 5 shows that students are asked to find some values more frequently than to prove propositions. The answer to a `Find` problem, `Find`$(v)[\phi(v)]$, is expected to be a characteristic function $f(v)$ that returns `true` if and only if v satisfies $\phi(v)$. The answer to a `Draw` problem `Draw`$(v)[\phi(v)]$ should be the geometric object v expressed as a characteristic function on $\mathbb{R}^2$ or $\mathbb{R}^3$. Figure 3 is an example of a `Find` problem taken from IMO 1982.

To use a problem set in the above format as benchmark data, we need a *rule* to judge whether a system's output is acceptable or not. It is clear for the `Show` directives: true or false. We regard a `Draw` directive as a variant of `Find` problems for which the system is supposed to find a formula that defines the geometric object. Then, what is "to solve a `Find` problem?" Roughly speaking, a solver is supposed to give a *correct* solution in its *simplest* form. We will discuss the properties an answer formula for a `Find` problem has to satisfy in Section 4.

2.3. *Representation language*

We formalized all the problems in a single theory on the basis of ZF regardless of their context. In formality, it is a typed lambda calculus with parametric polymorphism. This is again due to the fully automatic, end-

Table 4

Statistics on the syntactic properties (min/avg/max/**median**)

	Todai Robot Project Math Benchmark				TPTP-THF
	Ex	**Univ**	**IMO**	**All**	
# of formulae	1 / 2 / 9 / **2**	1 / 3 / 13 / **2**	1 / 2 / 25 / **1**	1 / 2 / 25 / **2**	1 / 103 / 5639 / **11**
# of atoms	15 / 86 / 554 / **71**	21 / 131 / 658 / **103**	11 / 75 / 359 / **64**	11 / 98 / 658 / **75**	1 / 724 / 63737 / **75**
Avg atoms per formula	10 / 40 / 138 / **34**	5 / 49 / 183 / **47**	4 / 58 / 325 / **52**	4 / 48 / 325 / **44**	0 / 21 / 811 / **4**
# of symbols	3 / 17 / 33 / **16**	6 / 20 / 34 / **20**	4 / 14 / 34 / **13**	3 / 17 / 34 / **16**	1 / 45 / 1442 / **9**
# of variables	1 / 11 / 54 / **8**	1 / 15 / 69 / **12**	0 / 8 / 39 / **7**	0 / 11 / 69 / **8**	0 / 152 / 11290 / **19**
# of λ	0 / 4 / 22 / **3**	0 / 4 / 23 / **3**	0 / 1 / 9 / **1**	0 / 3 / 23 / **2**	0 / 22 / 385 / **2**
# of $\forall$	0 / 2 / 49 / **0**	0 / 2 / 24 / **0**	0 / 5 / 24 / **4**	0 / 3 / 49 / **0**	0 / 122 / 10753 / **9**
# of $\exists$	0 / 5 / 38 / **3**	0 / 9 / 50 / **6**	0 / 2 / 20 / **1**	0 / 6 / 50 / **4**	0 / 8 / 496 / **2**
# of connectives	12 / 73 / 485 / **60**	17 / 111 / 494 / **85**	11 / 66 / 255 / **57**	11 / 83 / 494 / **65**	0 / 569 / 51044 / **53**
Max formula depth	8 / 20 / 50 / **19**	12 / 25 / 59 / **23**	9 / 21 / 49 / **20**	8 / 22 / 59 / **21**	2 / 36 / 359 / **11**
Avg formula depth	0 / 4 / 9 / **4**	0 / 4 / 9 / **4**	0 / 5 / 9 / **5**	0 / 4 / 9 / **4**	0 / 5 / 9 / **6**

```
;; <PROBLEM-TEXT>
;; The diagonals $AC$ and $CE$ of the regular hexagon $ABCDEF$ are
;; divided by the inner points $M$ and $N$, respectively, so that
;; ¥[
;;    ¥frac{AM}{AC} = ¥frac{CN}{CE} = r.
;; ¥]
;; Determine $r$ if $B$, $M$, and $N$ are collinear.
;; </PROBLEM-TEXT>
;;-------------------------------------------------------------
(def-directive problem_IMO_1982_2
  (Find (r)
    (exists (A B C D E F M N)
        (& (is-regular (polygon (list-of A B C D E F)))
           (on M (seg A C))
           (on N (seg C E))
           (~ (= M A)) (~ (= M C))
           (~ (= N C)) (~ (= N E))
           (= (/ (length-of (seg A M)) (length-of (seg A C)))
              (/ (length-of (seg C N)) (length-of (seg C E))))
           (= (/ (length-of (seg A M)) (length-of (seg A C)))
              r)
           (colinear B M N)))))

(def-answer problem_IMO_1982_2
   (lambda r (= r (/ 1 3))))
;;-------------------------------------------------------------
```

Fig. 3. Problem file example (IMO 1982, Problem 5).

Table 6

Logical translations in ZF set theory and Peano arithmetic

a) There are two prime numbers less than 4.
ZF: $\lvert\{n \in \mathbb{N} \mid \mathrm{prime}(n) \wedge n < 4\}\rvert = 2$
PA$_1$: $\exists n_1 \exists n_2 \left(\begin{array}{l} \mathrm{prime}(n_1) \wedge \mathrm{prime}(n_2) \wedge \\ n_1 < 4 \wedge n_2 < 4 \wedge n_1 \neq n_2 \wedge \\ \forall m \left(\begin{array}{l} (\mathrm{prime}(m) \wedge m < 4) \\ \rightarrow (m = n_1 \vee m = n_2) \end{array}\right) \end{array}\right)$
PA$_2$: number_of(prime_less_than(4)) $= 2$
b) There is an even number of prime numbers less than 4.
ZF: $\exists k \in \mathbb{N}(\mathrm{even}(k) \wedge \lvert\{n \mid \mathrm{prime}(n) \wedge n < 4\}\rvert = k)$
PA: $\exists k(\mathrm{even}(k) \wedge \text{number\_of}(\text{prime\_less\_than}(4)) = k)$
c) There are infinitely many prime numbers greater than 4.
ZF: $\lvert\{n \in \mathbb{N} \mid \mathrm{prime}(n) \wedge n > 4\}\rvert = \omega$
PA: $\neg\exists N \forall n((\mathrm{prime}(n) \wedge n > 4) \rightarrow n < N)$

to-end task setting. In informal mathematics, a word sometimes means an object, sometimes a relation, and sometimes a higher-order function. When we refer to "the derivative of f," we treat f as if it were an object though it is indeed a function. Parametric polymorphism is utilized to have polymorphic lists and sets in the language and define various operations on them while keeping the axioms and the lexicon concise.

Table 6 also demonstrates why a higher-order language is appropriate as the target language. Mechanical translation assumes a systematic correspondence between the syntactic structures of the input and output languages. That is, the output language needs to have enough expressive power so that any two natural language sentences having the same syntactic structure can be translated into two logical formulas having at least similar syntactic structures. The results of the translations of a), b), and c) into ZF in Table 6 have the same or at least a similar structure thanks to the set builder notation such as $\{n \in \mathbb{N} \mid \mathrm{prime}(n) \wedge n > 4\}$, which is expressed using λ-abstraction in the library. However, the expressions of the three sentences in PA must be different. Although a conservative extension of the theory allows us to translate a) and b) in a similar way by introducing the functions such as 'prime_less_than' and 'number_of,' we cannot translate c) analogously since the concept of finiteness cannot be expressed in first-order logic. Meanwhile, the expressibility of ZF allows almost word-by-word translations for all sentences.

We believe the vast majority of our benchmark problems can be eventually expressed in first-order logic. To mechanically fill the gap between the heavy-duty language and the relatively simple content is however a mandatory step to connect natural language processing and automated reasoning together for end-to-end automatic problem solving.

Since our mechanical translator is still under development, the problems were formalized manually at the current stage. Operators, all majored in computer science and/or mathematics, were trained to translate the problems as faithfully as possible to the original natural language statements following the design of the translation module. The sets of new symbols and their defining axioms were introduced in parallel with the problem formalization, to match the problem formula as closely as possible to the problem text. All the formalized problems were reviewed by one of the developers of the translation module to reject a formalization that involves paraphrasing or re-interpretation of the problem that is not possible in the mechanical translation.

In the language, we currently have 59 types including those shown in Table 7. The types are somewhat redundant in that we can represent, e.g., `EqnR` (equation in domain $\mathbb{R}$) simply by a function of type $\mathtt{R} \to \mathtt{R}$ by regarding $f : \mathtt{R} \to \mathtt{R}$ as representing $f(x) = 0$. The abundance of types, however, helped a lot in organizing the axioms and debugging the formalized problems. Figure 4 presents an excerpt from an axiom file that includes two type definitions (two `def-preds`) and two axioms.

All in all, the language shall be understood as a conservative extension of ZF set theory. It thus has some overlap with previous efforts toward formalizing a large part of mathematics, such as Mizar's mathematics library [18]. However, some essential parts of the system (e.g., the definition of the real numbers and arithmetic) are left undefined. Instead of writing all the inference rules explicitly, we delegated computer algebra systems to take care of it. Although it is not within our current research focus, full formalization of the system (maybe by embedding it into an existing formalized mathematics library) is an interesting future direction.

Table 7
Types defined in the representation language

truth values	`Bool`
numbers	`Z` (integers), `Q` (rationals), `R` (reals), `C` (complex numbers)
polynomials	`Poly`
single variable functions	`R2R` ($\mathbb{R} \to \mathbb{R}$), `C2C` ($\mathbb{C} \to \mathbb{C}$)
single variable equations	`EqnR` (in domain $\mathbb{R}$), `EqnC` (in $\mathbb{C}$)
points in 2D/3D space	`2d.Point`, `3d.Point`
geometric objects	`2d.Shape`, `3d.Shape`
vectors and matrices	`2d.Vector`, `3d.Vector`
matrices	`2d.Matrix`, `3d.Matrix`
angles	`2d.Angle`, `3d.Angle`
number sequences	`Seq`
limit values of functions	`LimitVal`
polymorphic containers	`SetOf`(α), `ListOf`(α)
polymorphic tuples	`Pair`(α, β), `Triple`(α, β, γ)

2.4. Related work

Development of a well-designed benchmark is crucial in the research field of automated reasoning. The most notable example is the "Thousands of Problems for Theorem Provers" (TPTP) [44], which covers various domains and several problem formats including CNF, first-order formula with quantifiers, and typed higher-order logic. Previous efforts have also accumulated benchmarks for various branches of AR, such as the satisfiability problems in propositional logic (SAT) [20], satisfiability modulo theory (SMT) [6], inductive theorem proving [14], and geometry problems [37]. However, the current study is the first attempt to offer a large collection of curriculum mathematics problems including not only proof problems but also `Find` and `Draw` problems with a wide range of difficulties as a benchmark for AR and mechanized mathematics technologies.

Our benchmark data would be of special interest to computer algebra (CA) and SMT community, where decision and quantifier-elimination procedure for a formula involving non-linear real arithmetic is one of the major themes. As was shown in Section 1, our benchmark includes more than 500 problems expressible in the theory of RCF with a variety of difficulty. We plan to provide them also in the form of RCF formulas. It would be a good addition to the existing benchmark problems for CA and SMT because the uniformity of the existing benchmark problems is raised as an issue [15].

```
;; tangent(S1, S2, P) <-> geometric objects
;; S1 and S2 are tangent at point P
(def-pred
  tangent :: Shape -> Shape -> Point => Bool)

(axiom
  def_tangent_line_and_circle
  (p q c r P)
  (<-> (tangent (line p q) (circle c r) P)
       (& (on P (line p q))
          (perpendicular (line c P) (line p q))
          (= (distance^2 P c) (^ r 2)))))
```

```
;; maximum(S, m) <->
;; m is the maximum element of set S
(def-pred
  maximum :: (SetOf R) -> R => Bool)

(axiom
  def_maximum
  (set max)
  (<-> (maximum set max)
       (& (elem max set)
          (forall (v)
              (-> (elem v set)
                  (<= v max))))))
```

Fig. 4. Type definitions and axioms.

End-to-end problem solving of mathematical word problems is a classical topic in artificial intelligence [7,17]. A mathematical word problem is a problem stated in natural language wherein several numerical relations are described in a real-world scenario. It attracts much attention recently in the field of natural language processing [21,28,29,36,39,40,47]. However, these studies mainly target at solving primary school level arithmetic word problems. In their nature, primary school arithmetic questions are quantifier-free. Moreover they tend to include only $\wedge$ (and) as the logical connective. Automated reasoning for such problems is hence not of much interest from the viewpoint of mechanized mathematics.

3. Problem samplers

We provide several sample problems taken from the first release of the library.

> Let ℓ be the trajectory of $(t+2, t+2, t)$ for t ranging over the real numbers. O$(0, 0, 0)$, A$(2, 1, 0)$, and B$(1, 2, 0)$ are on a sphere S, centered at C(a, b, c). Determine the condition on a, b, c for which S intersects with ℓ.
>
> Hokkaido Univ., 2011, Science Course, 3 (2)

In the data set, the above problem is formalized as shown in Fig. 5. It is not difficult to obtain an equivalent formula in the language of first-order RCF by rewriting the predicates and functions using their defining axioms. However, it results in a formula including 22 variables and 22 atoms, that is way above the ability of existing RCF-QE solvers to deal with. It is not very surprising seeing that the time complexity of the most common implementations of RCF-QE is doubly exponential in the number of variables in a given formula [13] (though it is in theory doubly exponential in the number of quantifier alternations). We enhanced existing RCF-QE algorithms to overcome the difficulty. Fortunately, our prototype system successfully solved this problem. We will explain the enhancement in detail in Section 5.

> Consider $0 < \frac{|ax-y|}{\sqrt{1+a^2}} < \frac{2\sqrt{2}}{x+y}$ for $x > 0$ and $y > 0$. Prove that there are only finitely many pairs of positive integers (x, y) that satisfy the above inequalities when a is a rational number.
>
> **Ex**, Math 3+C, Problem 09CBCE011

In the data set, the above problem is formalized as follows:

$$\forall a \in \mathbb{Q}\, \exists n \in \mathbb{Z}\big(n > 0 \wedge n = |S|\big)$$

where

$$S = \big\{(x, y) \in \mathbb{Z}^2 \mid P\big(\texttt{to\_real}(x), \texttt{to\_real}(y)\big)\big\}$$

and $P(x, y) \equiv (x > 0 \wedge y > 0 \wedge 0 < \frac{|ax-y|}{\sqrt{1+a^2}} < \frac{2\sqrt{2}}{x+y})$. Several $\in$'s preceding to domain names in the formula signify their types. Despite the seeming mixture of reals, integers, and rational numbers, we can easily find an equivalent formula in the language of Peano arithmetic:

$$\forall b \forall c \exists X \exists Y \forall x \forall y \begin{pmatrix} (c \neq 0 \wedge Q(x, y)) \\ \to (x < X \wedge y < Y) \end{pmatrix}$$

```
;; FILE: Univ-Hokkaido-2011-Ri-3.lsp
(def-directive
  hokudai_2011_Ri_3_2
   (Find (abc)
     (exists (a b c O A B C l S)
       (& (= abc (list-of a b c))
          (line-type l)
          (= l (shape-of-cpfun (lambda p (exists (t) (= p (point (+ t 2) (+ t 2) t))))))
          (sphere-type S)
          (= O (point 0 0 0)) (= A (point 2 1 0)) (= B (point 1 2 0))
          (on O S) (on A S) (on B S)
          (= C (point a b c))
          (= C (center-of S))
          (intersect l S)))))))
```

Fig. 5. Hokkaido University, 2011, Science Course, Problem 3 (2).

where all variables are in domain $\mathbb{Z}$ and

$$Q(x, y) \equiv x > 0 \wedge y > 0 \wedge bx - cy \neq 0 \\ \wedge (bx - cy)^2 (x + y)^2 < 8\bigl(b^2 + c^2\bigr).$$

In the course of the reformulation, we set $a = b/c$ and paraphrased the finiteness of the set of the lattice points (x, y) with the existence of the upper bounds of x and y. The mechanization of processes such as this is one of our ongoing research topics.

> Let $n \geqslant 3$ be an integer, and let $a_2, a_3, \dots, a_n$ be positive real numbers such that $a_2 a_3 \cdots a_n = 1$. Prove that $(1 + a_2)^2 (1 + a_3)^3 \dots (1 + a_n)^n > n^n$.
>
> IMO 2012, Problem 2

In the data set, this problem is formalized using a higher-order function `prod_from_to`, which is of type `(Z → R) → Z → Z → R` and corresponds to $\Pi_{\text{from}}^{\text{to}}$ in the common notation. This problem apparently requires some kind of inductive reasoning but the domain includes both real numbers and integers. Problems of this type are abundant in curriculum mathematics. We believe they will prove to be new and interesting and challenging problems for automated inductive reasoning, both theoretically (e.g., formalizing them in a suitable local theory other than ZF) and practically.

> S is the set $\{1, 2, 3, \dots, 1000000\}$. Show that for any subset A of S with 101 elements we can find 100 distinct elements x_i of S, such that the sets $\{a + x_i \mid a \in A\}$ are all pairwise disjoint.
>
> IMO 2003, Problem 1

It is straightforward to translate the above-mentioned problem in ZF:

$$\forall A\bigl(A \subset S \wedge |A| = 101 \rightarrow \phi_{100}(A)\bigr)$$

where $S = \{n \in \mathbb{N} \mid 1 \leqslant n \leqslant 1000000\}$ and

$$\phi_k(A) \\ = \exists X \begin{pmatrix} X \subset S \wedge |X| = k \wedge \\ \text{pairwise\_disjoint}(\ \{\{a + x \mid a \in A\} \mid \\ x \in X\}) \end{pmatrix}.$$

A reference answer to the problem proves the claim by first showing $\phi_1(A)$ and then $\phi_k(A) \rightarrow \phi_{k+1}(A)$ for $k = 1, \dots, 99$, assuming $A \subset S \wedge |A| = 101$.

The problem can be expressed in Presburger arithmetic as follows:

$$\forall a_1 \dots \forall a_{101} \left(\begin{array}{l} \bigwedge_{i=1}^{101} a_i \in S \wedge \bigwedge_{i \neq j} a_i \neq a_j \\ \rightarrow \exists x_1 \dots \exists x_{100} \begin{pmatrix} \bigwedge_{i=1}^{100} x_i \in S \wedge \\ \bigwedge_{i \neq j} x_i \neq x_j \wedge \\ \bigwedge_{i,j,k \neq l} a_i + x_k \neq a_j + x_l \end{pmatrix} \end{array} \right).$$

It may be possible to derive this formula from its original formulation in ZF with some heuristics. However, the reformulation would do little help in solving it since it includes more than 50 million atoms.

4. What constitutes an answer to a find problem?

Section 2.2 provided a brief discussion on the properties an answer formula for a `Find` problem has to satisfy for it to be regarded as acceptable (correctness and simplicity). Now we will discuss these in detail. In [45], Sutcliffe et al. proposed the conditions which answers of answer-extraction problems have to satisfy. Our definition of 'answer' encompasses theirs in spirit and covers more complicated cases beyond the extraction of a finite number of answers.

The definition of the *correctness* of an answer is straightforward. Given a problem $\texttt{Find}(x)[\psi(x, \boldsymbol{p})]$, where $\boldsymbol{p}$ stands for zero or more free parameters, an answer formula $\phi(x, \boldsymbol{p})$ must satisfy:

$$\forall x \forall \boldsymbol{p}\bigl(\psi(x, \boldsymbol{p}) \leftrightarrow \phi(x, \boldsymbol{p})\bigr). \tag{1}$$

An example of a correct answer formula $\phi'(x, \boldsymbol{p})$ is provided for each `Find` problem in the library. If $\phi'(x, \boldsymbol{p})$ is used instead of $\psi(x, \boldsymbol{p})$, the proof task for (1), which checks the correctness of the answer, should generally be easy.

The *simplicity* of an answer is harder to define. Suppose that you are given a problem,

$$\texttt{Find}(v : \mathbb{R})\bigl[v^2 = a\bigr],$$

in a mathematics test. Then, $\lambda v.(v^2 = a)$ is of course not an acceptable answer. Test-takers are expected to answer, for example,

$$\lambda v.\bigl(\bigl(a \geqslant 0 \wedge v = a^{1/2}\bigr) \\ \vee \bigl(a \geqslant 0 \wedge v = -a^{1/2}\bigr)\bigr).$$

An answer to a problem asking to find all real numbers v satisfying a formula $\phi(v)$ in the first-order language of RCF is called *simple* when it is in the form $\lambda v.\psi(v)$ satisfying the following conditions.

- $\psi(v)$ is a quantifier-free formula in disjunctive normal form, and
- each dual clause in $\psi(v)$ consists of atoms of the form of $v\ \rho\ \alpha$ (the answer) or $\beta\ \rho\ 0$ (conditions on the free parameters), where $\rho \in \{=, <, >, \leqslant, \geqslant\}$, and α and β are first-order terms not including v and comprises numbers, variables (i.e., parameters) and functions in $\{+,\ -,\ \cdot,\ /,\ \hat{}\ \text{(power)}\}$.

When a problem includes a condition on the free parameter(s), such as in

$$\mathtt{Find}(x : \mathtt{R})\big[p > 0 \wedge x^2 = p\big],$$

the answer must repeat the condition:

$$\lambda x.\big(p > 0 \wedge (x = \sqrt{p} \vee x = -\sqrt{p})\big).$$

It is necessary for mechanically checking the correctness of an answer by proving the equivalence to the correct answer.[5]

The aforementioned syntactic conditions for a problem classified in RCF should be acceptable because RCF allows quantifier elimination [46]. Furthermore, the statistics tell us that almost all pre-university math problems have explicit solutions (i.e., in the form of $x = \alpha, \beta > x > \gamma$, etc.)

For problems other than those expressible in RCF, we tried our best to capture a loose, common understanding in the form of acceptable answers by examining the model answers (for humans) to the benchmark problems. Our tentative definition of 'simple answers' is as follows:

- Simplicity of the sub-language: an answer formula should be in a language consisting of Boolean connectives, equality and inequalities, numbers, variables, and the four arithmetic operations and power calculations, sin, cos, tan, exp, log, and *a minimal use of lambda abstractions and quantifications*.
- Explicitness: whenever possible within the above restriction imposed on the language, the answer to a problem of the form $\mathtt{Find}(x)[\phi(x)]$ should be given using atoms such as $x = \alpha$ and $x > \alpha$, where α does not include x.

Note that we need quantification in general unless the problem is expressible in a theory that allows quantifier elimination. For instance, in the sub-language defined above, there is no way to express the answer to "Determine all positive numbers v that are divisible by three and also by two," other than, e.g.,

$$\exists k(v = 6k \wedge k > 0).$$

As for "*minimal use of* $\lambda, \forall, \exists$", we define the preference of answer form tentatively (Table 8). The hierarchy is our own creation but we tried our best to capture the conventions in the standard textbooks and problem books. The answer-check routine compares a solver's answer and the model answer in the data set, and checks whether the solver's answer ranks equal (or higher) in the hierarchy.

5. Prototype solver

While developing the benchmark data set, we also developed a prototype mathematical problem solver system (overviewed in Fig. 6). Given a formalized problem, the system first rewrites it iteratively using the axioms and several equivalence-preserving transformation rules such as beta-reduction, extensional equality between functions:

$$\lambda x.M = \lambda x.N \Leftrightarrow \forall x(M = N),$$

and variable elimination by substitution:

$$\forall x\big(x = f \rightarrow \phi(x)\big) \Leftrightarrow \phi(f),$$
$$\exists x\big(x = f \wedge \phi(x)\big) \Leftrightarrow \phi(f)$$

where x does not occur free in f. In the course of the rewriting process, several types of terms, such as multiplication and division of polynomials and integration, are evaluated (simplified) by computer algebra systems (CASs). Specifically, we use Mathemat-

[5]More precisely, the condition on the free parameter in the answer may be stronger (or weaker) than that given in the problem. For instance, to the problem

$$\mathtt{Find}(x : \mathtt{R})\big[p > 0 \wedge x^2 = p \wedge |x| > 1\big]$$

the possible solutions include

$$\lambda x.\big(p > 1 \wedge (x = \sqrt{p} \vee x = -\sqrt{p})\big)$$

as well as

$$\lambda x.\big(p > 0 \wedge \big((x = \sqrt{p} \wedge x > 1) \vee (x = -\sqrt{p} \wedge x < -1)\big)\big).$$

We allow both because they conform to the syntactic condition (after the distribution of $\wedge$) and are equivalent to the problem formula.

Table 8

Preference hierarchy on answer form

Directive type	Syntactic condition on the answer formula
$\texttt{Find}(v : \mathtt{R})[\phi(v, \boldsymbol{p})]$	1. $\bigvee_i(\bigwedge_j(v\ \rho_{ij}\ \alpha_{ij}) \wedge \psi_i(\boldsymbol{p}))$
	2. $\bigvee_i(\bigwedge_j f_{ij}(v)\ \rho_{ij}\ 0 \wedge \psi_i(\boldsymbol{p}))$
	3. $\bigvee_i(\exists(n : \mathtt{Z}).(\bigwedge_j(v\ \rho_{ij}\ \alpha_{ij}(n)) \wedge (n\ \rho_i\ \gamma_i)) \wedge \psi_i(\boldsymbol{p}))$
	4. $\bigvee_i(\exists(r : \mathtt{R}).(\bigwedge_j(v\ \rho_{ij}\ \alpha_{ij}(r)) \wedge (r\ \rho_i\ \gamma_i)) \wedge \psi_i(\boldsymbol{p}))$
$\texttt{Find}(v : \mathtt{Z})[\phi(v, \boldsymbol{p})]$	1. $\bigvee_i(\bigwedge_j(v\ \rho_{ij}\ \alpha_{ij}) \wedge \psi_i(\boldsymbol{p}))$
	2. $\bigvee_i(\exists(n : \mathtt{Z}).(v = \alpha_i(n) \wedge (n\ \rho_i\ \gamma_i)) \wedge \psi_i(\boldsymbol{p}))$
$\texttt{Find}(v : \texttt{SetOf(Point)})[\phi(v, \boldsymbol{p})]$	$\bigvee_i(v = \{(x, y) \mid \xi_i(x, y)\} \wedge \psi_i(\boldsymbol{p}))$

($\rho_* \in \{=, <, \leqslant, \geqslant, >\}$; α_*, $\alpha_*(\cdot)$, γ_*, $\xi_*(\cdot,\cdot)$: first-order terms not including v, x, y; $f_{ij}(v)$: first-order term; $\psi_i(\boldsymbol{p})$: quantifier-free first-order formula)

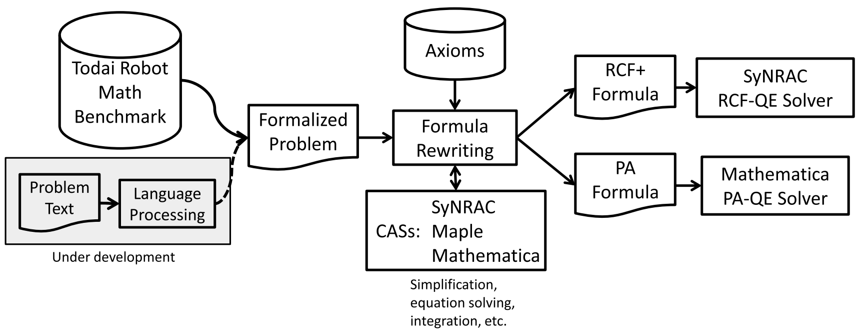

Fig. 6. System overview.

ica 10 for the operations on polynomials and Maple 18 for the integration and differentiation. Once the input is rewritten to a formula in the language of RCF, quantifier-elimination (QE) algorithms are invoked; we utilized the RCF-QE algorithm implemented in SyNRAC [24]. When QE is proceeded successfully, the remaining tasks, solving equations and inequalities in many cases, will be taken care by Mathematica. When the input is rewritten in the language of PA, we apply the `Reduce` command of Mathematica.

The prototype solver can thus be regarded as an enhancement of CASs with a much richer input language, which is as rich as allowing a translation of a mathematical problem stated in natural language on a word-by-word basis. The formula rewriting process is analogous to a compiler (e.g., from C++ to assembler) whose target languages are RCF and PA. It is however not guaranteed that the formula rewriting process produces a translation in the target languages since the input language can represent a far more wider variety of problems that fall outside of the expressiveness of RCF and PA. Furthermore, we need to carefully design the axioms so that they produce a computationally feasible reformulation of the concepts expressed by the predicate and function symbols.

As mentioned in Section 3, the first-order formulas generated by mechanical translation are much larger than expected [32]. We enhanced the RCF-QE algorithms by numerous techniques to handle them: choice of the computation order of sub-formulas [23,27], specialized QE algorithms for restricted input formulas [16], simplification of the intermediate formulas by utilizing the interim results [23], and so on.

Moreover we developed an approach to pre-process an RCF formula into a form more suitable for QE [22]. Specifically, we developed three methods to simplify the formulas based on the geometric invariance of the problems under translation, rotation, and scaling. On our benchmark, the simplifications were applicable to over 20% of the problems and provided substantial performance improvements. Additionally, we developed an algorithm for computing *the area enclosed by a set of curves* based on an enhancement of the cylindrical algebraic decomposition (CAD) algorithm.

We also implemented an extended RCF-QE procedure that reduces some of the problems involving trigonometric functions to RCF-QE problems. The reduction is applicable to a problem in a form such as

$$\exists x_1 \exists x_2 \dots \exists x_k \begin{pmatrix} \bigwedge_i \alpha_i < x_i < \beta_i \\ \wedge\, \phi(\sin(\sum_i m_i x_i), \cos(\sum_i n_i x_i)) \end{pmatrix}$$

where α_is and β_is are constants, m_is and n_is are integers, $\phi(s, c)$ is an RCF formula that does not include free occurrences of x_is. We also require that the condition $\alpha_i < x_i < \beta_i$ (modulo 2π) can be expressed sub-algebraically in terms of $\sin x_i$ and $\cos x_i$ (we store such combinations of α_i and β_i in a precompiled database). By expanding $\sin(\sum_i m_i x_i)$ and $\cos(\sum_i n_i x_i)$ into polynomials of $\sin x_i$ and $\cos x_i$ ($i = 1, \dots, k$), replacing $\sin x_i$ and $\cos x_i$ with new variables s_i and c_i, and adding $c_i^2 + s_i^2 = 1$ as new conditions, we have an equivalent RCF formula. For instance, we can convert

$$\exists x_1 \exists x_2 \begin{pmatrix} 0 < x_1 < \pi \wedge 0 < x_2 < \frac{\pi}{2} \\ \wedge\, y = \cos(x_1 + x_2) + \sin(x_1 - x_2) \end{pmatrix}$$

into

$$\begin{matrix} \exists c_1 \exists s_1 \\ \exists c_2 \exists s_2 \end{matrix} \begin{pmatrix} s_1 > 0 \wedge c_2 > 0 \wedge s_2 > 0 \\ \wedge\, y = c_1 c_2 - s_1 s_2 + s_1 c_2 - c_1 s_2 \\ \wedge\, c_1^2 + s_1^2 = 1 \\ \wedge\, c_2^2 + s_2^2 = 1 \end{pmatrix}.$$

This conversion is applicable only to a problem in quite a limited form. It however covers a large number of trigonometric problems in pre-university problems. MetiTarski [1] and Polya [5] are special-purpose provers developed for decision problems involving trigonometric and other transcendental functions. They mainly target at proving (Boolean combinations of) inequalities between transcendental functions. Although they can handle various types of transcendental functions, it is not straightforward to apply them to `Find` problems since we need not only to verify a closed formula but also to eliminate quantified variables to find an answer.

6. Experiments

The prototype system was run on the benchmark problems with a time limit of 600 seconds per problem (including the time spent on checking the correctness of the answers). Table 9 shows the number of successfully solved problems, minimum, median, average, and maximum (wallclock) time spent on solved problems, number of failures due to timeout, wrong answers (disproofs for `Show` or wrong answers for `Find` or `Draw` directives), and those not solved due to various reasons (the column headed ‘Other’). The machine used for the experiments is a 64-bit machine with 2.6 GHz AMD Opteron processors and 130 GB of RAM. Approximately one-third of the ‘Other’ cases were due to a failure in the problem reformulation phase; i.e., for those problems, the system could not find an equivalent formula expressible in either RCF or PA. Wrong answers were due to known bugs in our formula rewriting module.

Overall, the performances for the **Ex**, **Univ**, and **IMO** divisions seem to well reflect the inherent differences in their difficulty levels. Success rates on the RCF problems in **Ex** and **Univ** division are quite close, but the solver spent more time in solving **Univ** problems. The **IMO** problems are apparently more difficult than **Ex** and **Univ** problems as can be seen in the much lower success rate on RCF and PA problems and the higher timeout rate on the RCF problems. It is not surprising but justifies our primary motivation behind the development of the benchmark library: *we can gauge the ability of a mechanized reasoning system using pre-university students’ ability as the reference points.*

In comparison with our past experimental results [33], the speed of the computation and the rate of successfully solved problems are improved. For instance, on **Univ** problems, the median of the computation time was improved from 26.5 sec to 10.0 sec per problem and the rate of successfully solved problems was improved from 58.0% to 68.3%. The improvement in the speed is mainly due to the refinement of the RCF-QE procedure (Section 5). The improvement in the rate of success is brought by both the speed-up and the fact that more problems are successfully reformulated in RCF or PA by the axioms added for more comprehensive definitions of the predicates and function symbols (i.e., definitions for various special cases, for which general definitions cannot be formulated in RCF nor PA, e.g., ‘length of a curve’ vs. ‘length of the perimeter of a triangle’).

Table 10, Table 11, and Table 12 show further analysis of the results obtained for the three divisions. Table 10 lists the performance figures for the RCF problem subsets in the **Ex** division that are rated level 1 to 5 in the exercise books. The rating was done by the editors of the exercise books: level 1 to 3 signify textbook exercise level and level 4 and 5 signify university entrance exam level. We see a clear difference between those rated level 1 or 2, and 4 or 5, especially in the percentages of the problems that had a timeout.

Table 9

Overall results

		Succeeded		Failed		
		Success %	Time (sec)	Timeout	Wrong	Other
			Min / Med / Avg / Max			
Ex	RCF	68.4% (119/174)	2 / 5.0 / 25.7 / 529	21.3%	1.1%	9.2%
	PA	48.8% (41/ 84)	2 / 3.0 / 4.3 / 64	14.3%	0.0%	36.9%
	Other	10.0% (3/ 30)	3 / 4.0 / 165.3 / 489	3.3%	3.3%	83.3%
	All	56.6% (163/288)	2 / 4.0 / 22.9 / 529	17.4%	1.0%	25.0%
Univ	All (RCF only)	68.3% (166/243)	3 / 10.0 / 33.0 / 598	20.2%	2.9%	8.6%
IMO	RCF	23.3% (27/116)	3 / 16.0 / 36.0 / 226	66.4%	0.9%	9.5%
	PA	9.5% (4/ 42)	4 / 17.5 / 136.5 / 507	14.3%	0.0%	76.2%
	Other	5.6% (3/ 54)	2 / 3.0 / 6.0 / 13	5.6%	0.0%	88.9%
	All	16.0% (34/212)	2 / 11.5 / 45.2 / 507	40.6%	0.5%	42.9%

Table 10

Breakdown of results on **Ex** RCF problem by number of stars

	Succeeded		Failed		
# of stars	Success %	Time (sec)	Timeout	Wrong	Other
		Min / Med / Avg / Max			
1	82.4% (28/34)	2 / 4.0 / 8.6 / 99	8.8%	0.0%	8.8%
2	73.5% (25/34)	3 / 5.0 / 29.3 / 529	5.9%	2.9%	17.6%
3	72.7% (24/33)	3 / 6.0 / 43.3 / 502	21.2%	0.0%	6.1%
4	60.5% (23/38)	3 / 6.0 / 32.9 / 477	23.7%	2.6%	13.2%
5	54.3% (19/35)	3 / 11.0 / 15.5 / 37	45.7%	0.0%	0.0%

Table 11

Breakdown of results on **Univ** RCF problems by university

University	# of all problems	RCF problems %	Overall success %	Success % on RCF problems
Hokkaido	72	43.1% (31/72)	30.6% (22/72)	71.0% (22/31)
Tohoku	80	51.2% (41/80)	37.5% (30/80)	73.2% (30/41)
Tokyo	160	37.5% (60/160)	21.9% (35/160)	58.3% (35/60)
Nagoya	72	41.7% (30/72)	23.6% (17/72)	56.7% (17/30)
Osaka	64	37.5% (24/64)	31.2% (20/64)	83.3% (20/24)
Kyoto	88	42.0% (37/88)	34.1% (30/88)	81.1% (30/37)
Kyushu	96	36.5% (35/96)	26.0% (25/96)	71.4% (25/35)

Table 12

Results for **IMO** problems by decade

Years	Human score %	Machine score %	Success %	Failed		
				Timeout	Wrong	Other
1959–69	58.23%	25.94%	32.1% (18/56)	42.9%	0.0%	25.0%
1970–79	46.57%	8.50%	16.7% (5/30)	40.0%	0.0%	43.3%
1980–89	44.35%	7.41%	12.5% (4/32)	37.5%	0.0%	50.0%
1990–99	38.27%	5.00%	8.3% (3/36)	30.6%	2.8%	58.3%
2000–13	34.31%	4.76%	7.4% (4/54)	48.1%	0.0%	44.4%

Table 11 lists the performance figures for each university from which the exam problems were taken. Although average scores etc. of the entrance exams are not published, a statistical analysis undertaken by major prep schools tells us that the average score of successful applicants to the top universities is around 30-60% depending on schools and departments. Hence, it is very plausible that a machine will come to have the ability to pass the entrance mathematics exams of top universities if it is able to cover areas other than RCF.

Finally, Table 12 lists the results on **IMO** problems taken from different time periods. Human and Machine percentage scores in the table shows the ratio between the attained points (by all contestants in a year and by our system, respectively) and all possible points.[6] It seems that the IMO problems are getting harder year by year not only for human participants but more so for our system.

We believe that these experimental results support our decision on the library organization, and encourage us to further proceed toward the goal of end-to-end math problem solving with the monolithic logical language based on ZF.

7. Conclusion and prospects

In this paper, we introduced a benchmark problem library for mechanized mathematics technologies. The library consists of curriculum mathematics problems taken from exercise problem books, university entrance exams, and International Mathematical Olympiads. Unlike other existing benchmark libraries, this one contains problems that are formalized so that they are obtainable as the result of mechanical translation of the original problems expressed in natural language. Preliminary experimental results we obtained for our prototype system on the benchmark show that its performance is comparable to that of candidates for admission to top universities, at least for problems in real-closed fields.

As can be seen in the example problems and the experimental results, the benchmark library offers a challenge to various areas of mechanized mathematics and calls for an integration of these technologies. They include (but not limited to): enhancement of basic reasoning algorithms such as quantifier elimination procedures (e.g., [9–11,23,24,27,43]), simplification of a problem by exploiting the symmetry in the problem [2–4,19], inductive theorem proving beyond integer domain, and representation change of the problem across different local theories and formalisms (e.g., [26,34,35,38]).

Our future plan includes the expansion of the library with more problems on integer arithmetic, transcendental functions, combinatorics, and a mixture of real and integer arithmetics as well as development and performance improvement of the natural language processing module for an end-to-end system.

[6]The statistics were taken from the official IMO website: https://www.imo-official.org/results_year.aspx.

References

[1] B. Akbarpour and L.C. Paulson, MetiTarski: An automatic theorem prover for real-valued special functions, *Journal of Automated Reasoning* **44**(3) (2010), 175–205. doi:10.1007/s10817-009-9149-2.

[2] N.H. Arai, Tractability of cut-free Gentzen type propositional calculus with permutation inference, *Theoretical Computer Science* **170**(1) (1996), 129–144. doi:10.1016/S0304-3975(96)80704-3.

[3] N.H. Arai and R. Masukawa, How to find symmetries hidden in combinatorial problems, in: *Proceedings of the Eighth Symposium on the Integration of Symbolic Computation and Mechanized Reasoning*, 2000.

[4] R. Atkey, P. Johann and A. Kennedy, Abstraction and invariance for algebraically indexed types, in: *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL '13*, ACM, New York, NY, USA, 2013, pp. 87–100, http://doi.acm.org/10.1145/2429069.2429082.

[5] J. Avigad, R.Y. Lewis and C. Roux, A heuristic prover for real inequalities, *Journal of Automated Reasoning* **56**(3) (2016), 367–386. doi:10.1007/s10817-015-9356-y.

[6] C. Barrett, A. Stump and C. Tinelli, The Satisfiability Modulo Theories Library (SMT-LIB), 2010, www.SMT-LIB.org.

[7] D.G. Bobrow, Natural language input for a computer problem solving system, PhD thesis, Massachusetts Institute of Technology, 1964.

[8] J. Bos, Wide-coverage semantic analysis with boxer, in: *Semantics in Text Processing. STEP 2008 Conference Proceedings. Research in Computational Semantics*, J. Bos and R. Delmonte, eds, College Publications, 2008, pp. 277–286. doi:10.3115/1626481.1626503.

[9] R. Bradford, J.H. Davenport, M. England, S. McCallum and D. Wilson, Truth table invariant cylindrical algebraic decomposition, *Journal of Symbolic Computation* **76** (2016), 1–35. doi:10.1016/j.jsc.2015.11.002.

[10] C.W. Brown, Open non-uniform cylindrical algebraic decompositions, in: *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation. ISSAC '15*, ACM, New York, NY, USA, 2015, pp. 85–92. doi:10.1145/2755996.2756654.

[11] C. Chen and M. Moreno Maza, Quantifier elimination by cylindrical algebraic decomposition based on regular chains, *Journal of Symbolic Computation* **75**(C) (2016), 74–93. doi:10.1016/j.jsc.2015.11.008.

[12] S. Clark and J.R. Curran, Wide-coverage efficient statistical parsing with CCG and log-linear models, *Computational Linguistics* **33** (2007). doi:10.1162/coli.2007.33.4.493.

[13] J.H. Davenport and J. Heintz, Real quantifier elimination is doubly exponential, *Journal of Symbolic Computation* **5**(1–2) (1988), 29–35, http://www.sciencedirect.com/science/article/pii/S074771718880004X. doi:10.1016/S0747-7171(88)80004-X.

[14] L.A. Dennis, J. Gow and C. Schürmann, Challenge Problems for Inductive Theorem Provers v1.0, Technical report ULCS-07-004, University of Liverpool, Department of Computer Science, 2007.

[15] M. England and J.H. Davenport, Experience with heuristics, benchmarks & standards for cylindrical algebraic decomposition, in: *Proceedings of the 1st Workshop on Satisfiability Checking and Symbolic Computation Co-Located with 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2016)*, Timisoara, Romania, September 24, 2016, E. Ábrahám, J.H. Davenport and P. Fontaine, eds, CEUR Workshop Proceedings, Vol. 1804, CEUR-WS.org, 2016, pp. 24–31, http://ceur-ws.org/Vol-1804/paper-06.pdf.

[16] R. Fukasaku, H. Iwane and Y. Sato, Improving a CGS-QE algorithm, in: *Revised Selected Papers of the 6th International Conference on Mathematical Aspects of Computer and Information Sciences – Volume 9582. MACIS 2015*, Springer-Verlag Inc., New York, NY, USA, 2016, pp. 231–235.

[17] J.P. Gelb, Experiments with a natural language problem-solving system, in: *Proceedings of the 2nd International Joint Conference on Artificial Intelligence. IJCAI'71*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1971, pp. 455–462.

[18] A. Grabowski, A. Korniłowicz and A. Naumowicz, Mizar in a nutshell, *Journal of Formalized Reasoning* **3**(2) (2010), 153–245.

[19] J. Harrison, Without loss of generality, in: *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Proceedings*, Munich, Germany, August 17–20, 2009, 2009, pp. 43–59. doi:10.1007/978-3-642-03359-9_3.

[20] H.H. Hoos and T. Stützle, *SATLIB: An Online Resource for Research on SAT*, IOS Press, 2000, pp. 283–292.

[21] M.J. Hosseini, H. Hajishirzi, O. Etzioni and N. Kushman, Learning to solve arithmetic word problems with verb categorization, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP*, Doha, Qatar, October 25–29, 2014, A meeting of SIGDAT, a Special Interest Group of the ACL, 2014, pp. 523–533, http://aclweb.org/anthology/D/D14/D14-1058.pdf. doi:10.3115/v1/D14-1058.

[22] H. Iwane and H. Anai, Formula simplification for real quantifier elimination using geometric invariance, in: *Proceedings of the 42nd International Symposium on Symbolic and Algebraic Computation (ISSAC-2017)*, 2017, to appear.

[23] H. Iwane, T. Matsuzaki, N. Arai and H. Anai, Automated natural language geometry math problem solving by real quantier elimination, in: *Proceedings of the 10th International Workshop on Automated Deduction (ADG2014)*, 2014, pp. 75–84.

[24] H. Iwane, H. Yanami, H. Anai and K. Yokoyama, An effective implementation of symbolic-numeric cylindrical algebraic decomposition for quantifier elimination, *Theor. Comput. Sci.* **479** (2013), 43–69. doi:10.1016/j.tcs.2012.10.020.

[25] C. Kaliszyk, G. Sutcliffe and F. Rabe, TH1: The TPTP typed higher-order form with rank-1 polymorphism, in: *Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning (PAAR 2016)*, 2016, pp. 41–55.

[26] M. Kerber and M. Pollet, A tough nut for mathematical knowledge management, in: *Mathematical Knowledge Management*, Springer, 2006, pp. 81–95. doi:10.1007/11618027_6.

[27] M. Kobayashi, H. Iwane, T. Matsuzaki and H. Anai, Efficient subformula orders for real quantifier elimination of non-prenex formulas, in: *Mathematical Aspects of Computer and Information Sciences – 6th International Conference, MACIS 2015, Revised Selected Papers*, Berlin, Germany, November 11–13, 2015, pp. 236–251. doi:10.1007/978-3-319-32859-1_21.

[28] R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni and S. Ang, Parsing algebraic word problems into equations, *Transactions of the Association for Computational Linguistics* **3** (2015), 585–597, https://transacl.org/ojs/index.php/tacl/article/view/692.

[29] N. Kushman, Y. Artzi, L. Zettlemoyer and R. Barzilay, 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, 2014, pp. 271–281, http://www.aclweb.org/anthology/P14-1026. doi:10.3115/v1/P14-1026.

[30] T. Kwiatkowksi, L. Zettlemoyer, S. Goldwater and M. Steedman, Inducing probabilistic CCG grammars from logical form with higher-order unification, in: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2010, pp. 1223–1233.

[31] T. Matsuzaki, T. Ito, H. Iwane, H. Anai and N.H. Arai, Semantic parsing of pre-university math problems, in: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics*, 2017, to appear.

[32] T. Matsuzaki, H. Iwane, H. Anai and N.H. Arai, The most uncreative examinee: A first step toward wide coverage natural language math problem solving, in: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014, pp. 1098–1104.

[33] T. Matsuzaki, H. Iwane, M. Kobayashi, Y. Zhan, R. Fukasaku, J. Kudo, H. Anai and N.H. Arai, Race against the teens – benchmarking mechanized math on pre-university problems, in: *Automated Reasoning – 8th International Joint Conference*, IJCAR 2016, Coimbra, Portugal, June 27, N. Olivetti and A. Tiwari, eds, Proceedings. Lecture Notes in Computer Science, Vol. 9706, Springer, 2016, pp. 213–227, July 2, 2016.

[34] T. Matsuzaki, M. Kobayashi and N.H. Arai, An information-processing account of representation change: International mathematical olympiad problems are hard not only for humans, in: *Proceedings of the 38th Annual Conference of the Cognitive Science Society*, Cognitive Science Society, 2016, pp. 2297–2302.

[35] J. McCarthy, A tough nut for proof procedures, Technical report Stanford Artificial Intelligence Project Memo 16, Stanford University, 1964.

[36] A. Mitra and C. Baral, Learning to use formulas to solve simple arithmetic problems, in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 2144–2153.

[37] P. Quaresma, Thousands of geometric problems for geometric theorem provers (TGTP), in: *Automated Deduction in Geometry*, P. Schreck, J. Narboux and J. Richter-Gebert, eds, Lecture Notes in Computer Science, Vol. 6877, Springer, Berlin Heidelberg, 2011, pp. 169–181. doi:10.1007/978-3-642-25070-5_10.

[38] D. Raggi, A. Bundy, G. Grov and A. Pease, Automating change of representation for proofs in, *Discrete Mathematics (Extended Version). Mathematics in Computer Science* **10**(4) (2016), 429–457.

[39] S. Roy and D. Roth, Solving general arithmetic word problems, in: *Conference Proceedings – EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics (ACL), 2015, pp. 1743–1752. doi:10.18653/v1/D15-1202.

[40] S. Shi, Y. Wang, C.-Y. Lin, X. Liu and Y. Rui, Automatically solving number word problems by semantic parsing and reasoning, in: *EMNLP*, L. Màrquez, C. Callison-Burch, J. Su, D. Pighin and Y. Marton, eds, The Association for Computational Linguistics, 2015, pp. 1132–1142, http://dblp.uni-trier.de/db/conf/emnlp/emnlp2015.html#ShiWLLR15.

[41] S. Shuppan, 2014, Studyaid D.B. Chart-Shiki database (in Japanese).

[42] M. Steedman, *The Syntactic Process. Bradford Books*, MIT Press, Cambridge, 2001.

[43] Strzeboński, A., Cylindrical algebraic decomposition using local projections, *Journal of Symbolic Computation* **76**(C) (2016), 36–64. doi:10.1016/j.jsc.2015.11.018.

[44] G. Sutcliffe, The TPTP problem library and associated infrastructure: The FOF and CNF parts, v3.5.0, *Journal of Automated Reasoning* **43**(4) (2009), 337–362. doi:10.1007/s10817-009-9143-8.

[45] G. Sutcliffe, M. Stickel, S. Schulz, Urban and J. Answer, Extraction for TPTP, http://www.cs.miami.edu/~tptp/TPTP/Proposals/AnswerExtraction.html.

[46] A. Tarski, *A Decision Method for Elementary Algebra and Geometry*, University of California Press, Berkeley, 1951.

[47] L. Zhou, S. Dai and L. Chen, Learn to solve algebra word problems using quadratic programming, in: *EMNLP*, L. Màrquez, C. Callison-Burch, J. Su, D. Pighin and Y. Marton, eds, The Association for Computational Linguistics, 2015, pp. 817–822, http://dblp.uni-trier.de/db/conf/emnlp/emnlp2015.html#ZhouDC15.