Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

The Most Uncreative Examinee: A First Step toward Wide Coverage Natural Language Math Problem Solving

Takuya Matsuzaki,[1] **Hidenao Iwane,**[1,2] **Hirokazu Anai,**[1,2,3] **Noriko H. Arai**[1]
[1] National Institute of Informatics, Japan
[2] Fujitsu Laboratories Ltd., Japan [3] Kyushu University, Japan
{takuya-matsuzaki,arai}@nii.ac.jp; {iwane,anai}@jp.fujitsu.com

Abstract

We report on a project aiming at developing a system that solves a wide range of math problems written in natural language. In the system, formal analysis of natural language semantics is coupled with automated reasoning technologies including computer algebra, using logic as their common language. We have developed a prototype system that accepts as its input a linguistically annotated problem text. Using the prototype system as a reference point, we analyzed real university entrance examination problems from the viewpoint of end-to-end automated reasoning. Further, evaluation on entrance exam mock tests revealed that an optimistic estimate of the system's performance already matches human averages on a few test sets.

Introduction

Natural language math problem solving has a special status among various tests for the language understanding ability of a computational machinery. This is because one of the most successful approximations of our intuitive grasp of "the meaning of language" to date is to know the truth-conditions of the sentences that are usually expressed as logical formulas, and formal logic has been primarily developed for, and most successfully applied to, the formalization of mathematics.

We take a (much extended) conservative extension of Zermelo-Fraenkel (ZF) set theory as the target language of the translation from problem texts. ZF set theory is a natural choice because of its coverage over wide fields (as wide as almost all) of math, and a math problem written in natural language very often involves sets and related concepts such as tuples and functions, either implicitly or explicitly.

The downside of the expressiveness of the ZF set theory is that it is not practical to do reasoning directly based on it. We thus need to find another representation of the problem that is expressible in a more manageable theory through a mechanical procedure that operates on the initial ZF formula. It is a daunting task, in that it is apparently not possible in general, and it is a close kin to a long-standing problem in AI (McCarthy 1964).

Having it in mind, though, we have developed a prototype system, in which the initial ZF formula is iteratively rewritten by applying several kinds of equivalence-preserving transformation rules until we find the rewritten formula is directly re-interpretable in a theory that is more amenable to automatic reasoning. When it succeeds, we invoke a solver (e.g., a decision procedure or a prover) specialized to that theory to find the answer.

The simple design of our system is largely inspired by a classical result by Tarski (1951), which states that the theory of real closed field (RCF) allows quantifier-elimination (QE). From the viewpoint of problem solving, it means that, given a problem in the form of "Find the value of x," once we derive its representation as any formula ϕ in the first-order language of RCF, we can in principle find all possible values of x.

A serious limitation in the above approach is in the time complexity of RCF-QE. Currently, the most practical algorithm has the time complexity of doubly exponential in the number of variables in the input formula. The practical applicability of RCF-QE is hence very sensitive to how we represent the problem as an RCF formula. We are thus keen to know whether or not the above approach is practicable on a formula that is mechanically translated from natural language and then transformed by a simple rewriting algorithm.

We evaluated our prototype system on real university entrance exam problems. First we examined the problems from the viewpoint of end-to-end problem solving, with special care on the possibility of automatically identifying a mathematical theory in which a problem shall be solved. We then supplied the system with two sorts of translations of the problems in ZF set theory: one is manually translated and the other is semi-automatically derived from a linguistically annotated problem text. By comparing the results, we investigated the effect of the mechanical derivation of the ZF formulas on the representation change and the reasoning in a theory. Finally, we applied the system to several entrance exam mock tests, on which we can compare the system's performance with the average scores of the real test-takers. The experimental results are optimistic estimates of the system's end-to-end performance in that some of the language processing components are surrogated by linguistic annotations. The overall result is however encouraging and it suggests several directions of further investigations.

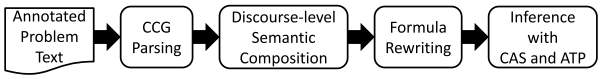

Figure 1: Problem solving pipeline

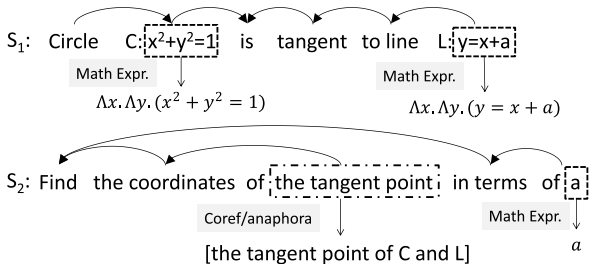

Figure 2: Linguistic annotation on the problem text

Overview of the Problem Solving Pipeline

We give an overview of the problem solving pipeline (Fig. 1). The system receives an annotated problem text as the input. The sentences in the problems are then translated to logical forms through grammar-based parsing, using the annotations on the text as the constraints in the parsing process. The sentence-level logical forms are then combined into a problem-level semantic representation according to the inter-sentence logical relations annotated on the problem. We then attempt to rewrite the the problem-level logical form into an equivalent formula that can be expressed in a theory in which automated reasoning is practically possible. Finally, a reasoner is invoked on the formula to find the answer. In the rest of this section, we first explain the linguistic annotations on the problems and then give some sketches of the processes in the pipeline.

Linguistic Annotations

We annotated the problem texts with four kinds of linguistic information: math expression semantics (Cramer, Koepke, and Schröder 2011), coreferential expressions (Haghighi and Klein 2010), syntactic dependency trees (Kudo and Matsumoto 2002), and logical connectives among sentences (Lin, Kan, and Ng 2009). Fig. 2 shows an example of an annotated problem that includes two sentences S_1 and S_2. In the figure, two math expressions in S_1 and one in S_2 are annotated with their semantic representations using the semantic language that will be introduced shortly. The words in the sentence are grouped into chunks[1] and the syntactic dependencies among the chunks are annotated (the arrows).

We also annotated coreferential expressions in the text with their referents. For instance, the definite noun phrase "the tangent point" in S_2 refers to an object that is not explicitly mentioned anywhere but understood from the context. To annotate various coreferential expressions uniformly, we annotated them with their paraphrases that identify the (understood) referents. For instance, "the tangent point" in S_2 is annotated with its paraphrase, "the tangent point of C and L," as shown in the figure.

Finally, we annotate the logical relations among the sentences as a binary tree whose leaves are the sentences and internal nodes are labeled with logical connectives. For the two sentences S_1 and S_2 in Fig. 2, we annotate their relation simply as S_1 & S_2, which stands for their conjunction.

The annotations are used as surrogates of the automatic language processing tools such as those cited above. We leave as future work the adaptation of existing language processing tools to the math text.

Semantic Representation

We use dynamic predicate logic (DPL) (Groenendijk and Stokhof 1991) as the base of our semantic representation language. The dynamic nature of the logic is essential for handling variable bindings across sentences. In a nutshell, existential quantifiers ($\exists$) in DPL have their scopes over dynamic conjunction (;) and implication ($\rightarrow$), but not across disjunction ($\vee$) nor negation ($\neg$). Furthermore, $\exists$-bindings across implications are interpreted as universal quantification. Thus, $(\exists x; P(x)); Q(x)$ and $(\exists x; P(x)) \rightarrow Q(x)$ are interpreted respectively (in the usual predicate logic notation) as $\exists x(P(x) \wedge Q(x))$ and $\forall x(P(x) \rightarrow Q(x))$.

We extend the language in a few directions. First, we use its higher-order version for representing mathematical functions and sets using lambda abstractions denoted as $(\Lambda x.M)$. We also use another kind of lambda abstractions $(\lambda x.M)$ for semantic composition through beta reduction. For example, the semantic representation of "the range of $y = x^2$ is $y \geq 0$" can be derived as follows:

$$
\begin{aligned}
&[\![\text{the range of } y = x^2 \text{ is } y \geq 0]\!] \\
&\rightarrow [\![\text{is}]\!]\ [\![y \geq 0]\!]\ ([\![\text{the range of}]\!]\ [\![y = x^2]\!]) \\
&\rightarrow (\lambda u.\lambda v.(v{=}u))\ [\![y \geq 0]\!]\ ([\![\text{the range of}]\!]\ [\![y = x^2]\!]) \\
&\rightarrow ([\![\text{the range of}]\!]\ [\![y = x^2]\!])\ =\ [\![y \geq 0]\!] \\
&\rightarrow (\lambda f.\Lambda y.(\exists w; y{=}f(w))\ (\Lambda x.x^2))\ =\ (\Lambda y.(y{\geq}0)) \\
&\rightarrow \Lambda y.(\exists w; y{=}(\Lambda x.x^2)w){=}\Lambda y.(y{\geq}0)
\end{aligned}
$$

The terms (and formulas) in the logic are typed as in the usual simply-typed lambda calculus. In addition to the type of truth values (`Bool`), we define basic types for real numbers (`R`), integers (`Z`), etc., and also polymorphic types such as lists of type α (`ListOf` α). These types are utilized both for organizing the knowledge-base (a large set of defining axioms for predicates and functions) and for the semantic disambiguation of the natural language sentences by the selectional restrictions of the words.

Finally, we add two operators for representing imperative sentences in the math problems. One denotes a request for a proof of a formula ϕ: **Show**$[\phi]$. The other denotes a request for finding values satisfying a condition $\phi(x)$: **Find**$(x)[\phi(x)]$. Hereafter, we call them *directives*. For instance, a problem: "Determine the range of c s.t. $x^2{+}x{+}c = 0$ has two real solutions" can be written as a **Find** directive:

$$
\mathbf{Find}(c)\left[\begin{array}{l} \exists x_1 \in \mathtt{R}; \exists x_2 \in \mathtt{R}; x_1 \neq x_2; \\ x_1^2 + x_1 + c = 0; x_2^2 + x_2 + c = 0 \end{array}\right].
$$

[1] The chunk is a syntactic unit called bun-setsu in Japanese. The English word chunks in the figure roughly match the bun-setsus in their Japanese translations.

Semantic Composition through CCG Parsing

The semantic representation of a sentence is composed using Combinatory Categorial Grammar (CCG) (Steedman 2001). In a CCG grammar, a handful of grammar rules are used to compose sentences from lexical items defined in a lexicon. A lexical item is a triple of a word, its category, and a semantic function. The category is either a basic category (S for sentence, NP for noun phrase, etc.) or a function category of the form X/Y or $X\backslash Y$, where X and Y are again categories. A word of category X/Y and $X\backslash Y$ takes its argument of category Y to form a phrase of category X. The direction of slashes specifies whether the argument phrase precedes ($\backslash$) the function word or succeeds ($/$) it. A grammar rule specifies how to combine the categories and the semantic functions in parallel. Two examples of such rules are forward application (>) and forward composition (>B):

$$\begin{array}{ll} X/Y : f + Y : x \to X : fx & (>) \\ X/Y : f + Y/Z : g \to X/Z : \lambda x.f(gx) & (>\mathrm{B}). \end{array}$$

The usage of "is" as in "x is 1" can hence be defined as:

$$\mathrm{is} := S\backslash NP/NP : \lambda y.\lambda x.(x = y),$$

which equates the subject (x) and the object (y) NPs.

We have implemented a Japanese CCG grammar. The overall design of the grammar follows the analysis by Bekki (2010). In addition to the basic constructions, we covered various phenomena including imperatives, focus particles with logical effects (similar to English "too", "only" etc.), it-clefts, and argument-taking nouns. Our current lexicon contains 6,000 lexical entries that define the associations between 1,600 word forms and 2,500 semantic functions.

We can in principle parse a sentence directly with the CCG grammar. In fact, wide-coverage CCG grammars and statistical parsers have been developed for several languages including Japanese and English (Clark and Curran 2007; Uematsu et al. 2013). They are however not directly applicable for our purpose because the semantic analyses produced by their grammars are not detailed enough, and the parsers are trained on newspaper text. We instead use a two-step parsing strategy where a dependency analysis is used as constraints on the structure of CCG parse trees, and a symbolic CCG parser searches for a coherent analysis obeying the constraints. In the experiments in the current paper, we hand-corrected the dependency analysis produced by a publicly available dependency parser (Kudo and Matsumoto 2002). We plan to retrain the parser on a dependency treebank on math problem texts that is currently under development.

Semantic Composition across Sentences

To handle a mixture of declarative sentences and imperative sentences in a math problem, we define the denotation of a discourse (i.e., a sequence of sentences) as a pair of formula D, which denotes the declarative content of the discourse, and a list of directives $[I_1, I_2, \dots]$, which represents the content of the imperative sentences. A complication comes in here because the declarative and imperative content of a discourse both depend on their preceding context.

For instance, the meanings of a noun phrase "the maximum value of x" and an imperative sentence "Find the value of x" cannot be fully determined in isolation from its context; the denotations of these expressions have to be defined as *functions* of the meanings of the preceding context. To derive the paired meanings of declarative and imperative content, we define two connectives & and $\Rightarrow$ as follows:

$$\begin{array}{l} [\![s_1 \& s_2]\!] = \lambda c.((D_1; D_2), I_1 +\!\!+ I_2) \\ [\![s_1 \Rightarrow s_2]\!] = \lambda c.((D_1 \to D_2), I_1 +\!\!+ I_2) \end{array}$$

where

$$\begin{array}{ll} D_1 = \pi_D([\![s_1]\!]c) & I_1 = \pi_I([\![s_1]\!]c) \\ D_2 = \pi_D([\![s_2]\!](c; D_1)) & I_2 = \pi_I([\![s_2]\!](c; D_1)), \end{array}$$

and π_D and π_I respectively stand for the projections from a pair to its first and second elements. The declarative (D_1) and imperative (I_1) content of s_1 may thus depend on s_1's preceding context c, which is passed as an argument of s_1's semantic function $[\![s_1]\!]$. In actuality, the context is just a dynamic conjunctions (;) of the declarative content of preceding sentences, which is accumulated along the logical structure among the sentences. The semantic function $[\![s_2]\!]$ hence receives the conjunction $(c; D_1)$ as its context regardless of its relation (& or $\Rightarrow$) to s_1.

Suppose we have the following lexical entries:

$$\begin{array}{l} \mathrm{assume} := S/S : \lambda s.\lambda c.(s(c), []) \\ \mathrm{find} := S/NP : \lambda t.\lambda c.(\top, \mathbf{Find}(z)[c; t(c) = z]). \end{array}$$

Assuming that the denotation[2] of "the solutions of $x^2 = a$" is $\lambda c.\Lambda x.(x^2 = a)$, we have the denotations of the two sentences, s_1: "Assume $a > 0$" and s_2: "Find the solutions of $x^2 = a$" as follows:

$$\begin{array}{l} s_1 : \lambda c.(a > 0, [\,]) \\ s_2 : \lambda c.(\top, \left[\mathbf{Find}(z)[c; (\Lambda x.(x^2 = a)) = z]\right]). \end{array}$$

By combining them according to the above definition, and then applying $[\![s_1 \& s_2]\!]$ to the empty context $\top$, we have :

$$[\![s_1 \& s_2]\!]\top = \left((a > 0; \top), \left[\mathbf{Find}(z)\left[\begin{array}{l} \top; a > 0; \\ (\Lambda x.(x^2 = a)) = z \end{array}\right]\right]\right).$$

Note that having $(a > 0; \top)$ in the first component is necessary since when there are more sentences that follow s_1 and s_2, the assumption $a > 0$ is still active there.

Formula Rewriting

The final output from the semantic composition is a formula in ZF set theory. Although the expressiveness and the uniformity of the semantic language is essential for understanding problems in various sub-domain with a single system, it is not practical to use the language directly as the vehicle of inference. We thus hope to find a formula that is equivalent to the initial input but can also be interpreted as a proposition expressed in a more manageable (especially decidable)

[2]The outer λc of $\lambda c.\Lambda x.(x^2 = a)$ actually does nothing since its meaning does not depend on the context, but context-dependent phrases such as "maximum of x" needs it to build its semantics.

theory. Although it is not possible in a general setting, we determined to start with a very simple procedure for that purpose. Our hope is, by gradually adding sophistications to the procedure, to reach a good characterization of the problems that are solvable by a modestly trained average human.

Our current procedure is a greedy (no backtracking) application of several types of equivalence-preserving rewriting rules. Once we find a rewritten formula is in a sub-language of ZF set theory that corresponds to the language of a local theory such as real-closed field or Peano arithmetic, we stop the rewriting and pass the formula to a solver (i.e., a decision procedure or a theorem prover).

We currently use rewriting rules of the following types:

- Trivial transformations of a formula, such as $\exists x(x{=}\alpha \wedge \phi(x)) \Rightarrow \phi(\alpha)$ and $\forall x(x{=}\alpha \rightarrow \phi(x)) \Rightarrow \phi(\alpha)$ (x is not free in α in both rules)
- β-reduction of the Λ-terms
- Rewriting of the terms by computer algebra systems (CASs): e.g., the calculation of the integrals and differentiations in the formula
- Rewriting of the predicates and functions using their definitions in a knowledge-base (axioms). E.g.,: $\mathrm{is\_divisible\_by}(n, m) \Rightarrow \exists k{\in}\mathsf{Z}(n = km)$ $\mathrm{distance}((x_1, y_2), (x_2, y_2)) \Rightarrow \sqrt{(x_1{-}x_2)^2 + (y_1{-}y_2)^2}$
- Evaluation of predicates and functions by a program implemented based on CASs

The last type of the rule is used for the predicates and functions for which direct definitions in the form axioms are impractical. For instance, a function in the semantic language is implemented to calculate the area of a region enclosed by a set of curves defined by polynomials.

Reasoning with Computer Algebra and ATP

The current implementation of our problem solving pipeline includes two solvers. One is a prover for the formulas on integer arithmetic that is implemented in the `Reduce` command of Mathematica 9. The other is our implementation of quantifier-elimination for real-closed field (RCF-QE). In this section, we give a brief overview of the RCF-QE solver.

The first-order language of RCF consists of polynomial equations and inequalities, boolean operators ($\vee, \wedge, \rightarrow, \neg$ etc.), and quantifiers ($\forall, \exists$). Together with a mechanical transformation from fractional roots and rational fractions of polynomials, we can express many mathematical problems in RCF. RCF allows quantifier-elimination (Tarski 1951): for any RCF formula ϕ, we can algorithmically find an equivalent quantifier-free formula ψ. It means that the theory of RCF is decidable (we get $\top$ or $\bot$ by QE on a closed formula), and also we can find the value (or the set of values) for x conditioned by an open formula $\phi(x)$ by solving a system of equations and inequalities resulted from QE on $\phi(x)$.

The time complexity of the most widely-used RCF-QE algorithm is doubly exponential in the number of variables in the formula (Collins 1975). Intensive research pursuing a practical QE algorithm has made much progress and resulted in sophisticated algorithms implemented in, e.g., REDLOG (Dolzmann and Sturm 1997), QEPCAD B (Brown 2003), and SyNRAC (Iwane et al. 2013). However, many interesting problems including more than six variables are still out-of-reach of the current QE algorithms.

To make things worse, the formulas produced by the mechanical translation tend to be ridiculously long. It is basically because of the redundancy in the mechanically derived formulas. Such redundancy is however inevitable since the translation is basically done word-by-word, without any (theory-dependent) reasoning based on the global context.

We thus enhanced the practical efficiency of our RCF-QE solver by introducing various techniques. They include (1) specialised algorithms for formulas in certain forms (Loos and Weispfenning 1993; Hong 1993; Iwane, Higuchi, and Anai 2013), (2) simplification of the intermediate formulas (Wilson, Bradford, and Davenport 2012; Dolzmann and Sturm 1995), (3) decomposition of the formulas into independently solvable parts, and (4) parallel computation.

Table 1: A classification of the problems

Theory		#Qs
Real Closed Field (RCF)		47
Peano Arithmetic (PA)		10
ZF Set Theory	Transcendental Func.	23
	RCF+PA	15
	Other	4
	total	99

Experiments on University Entrance Exam Math Problems

We applied our system to real university entrance exam math problems. We first show an analysis of the mathematical content of the problems from the viewpoint of automated reasoning embedded in an end-to-end problem solving pipeline. We then show experimental results based on two sorts of formal representations of the problems: one is translated manually from the problems and the other is derived semi-automatically from the linguistically annotated problem texts. We thus evaluated the additional difficulty due to the linguistic complexity of the problems.

Real Math Exams as a Benchmark for AR

The problems were taken from the entrance exams of seven national universities in Japan (Hokkaido Univ., Tohoku U., Tokyo U., Nagoya U., Osaka U., Kyoto U., and Kyushu U.) in year 1999. We chose year 1999 because a half of the more recent data has been used in the development of the system and the other half is kept for future use. In total, there were 124 problems in 65 parts in the 1999 exams. From the 124 problems, we eliminated 25 problems that involve probability theory or combinatorics because they are not directly translatable to our current semantic language.

The remaining 99 problems include various subjects such as real and natural number arithmetic, 2D and 3D geometry, linear algebra, precalculus, and calculus. Table 1 shows to what theories of mathematics they are to be classified. The problems classified in the first two rows had fairly clear signs

that indicate the main parts of the inference required in the problems can be framed either as a quantifier-elimination for RCF and equation solving, or a theorem proving in PA. They occupy approximately 60% of the problems.

The problems in the rest 40% are more problematic in that we do not know any theory in which the problems can be directly expressed and the reasoning can be carried out practically. We also note here that it is not always easy to mechanically determine whether or not a natural language math problem can be expressed in a certain theory. See Q1-Q3 below for instance:

Q1: Find the sides of the square whose perimeter is 4.
Q2: Find the perimeter of a circle whose radius is 1.
Q3: Find the maximum distance of two points on a circle whose radius is 1.

Despite the mutual overlaps in the concepts and wordings in them, Q1 and Q3 are expressible in RCF but Q2 is not. This is because Q2 essentially involves a transcendental number π. We thus know no matter how we represent the problem (e.g., by going back to the definition of integration), it is never expressible in the language of RCF.

Nonetheless, we give a rough classification of the problems in "ZF set theory" group as follows. First, 23 of the problems require some inference involving transcendental functions (trigonometric functions and their inverse, logarithm, and exponential). We found a half of them (12 problems) can be transformed to a formula in the language of RCF by using the functions of CASs (integration, max/min finding, and the calculation of limits) and replacement of transcendental functions with variables (e.g., $x := \sin\theta$, $y := \cos\theta$ with a constraint $x^2 + y^2 = 1$). The procedures are inevitably heuristic and we need further investigation to see to what extent we can automate it in practice.

The 15 problems labeled "RCF+PA" involve both natural numbers (or integers or rational numbers) and real numbers in the problems. Most of them require either a proof for a statement in the form of "... Show that the real number x is natural/rational number," or induction for proving a proposition on the mixture of natural and real numbers.

For the last four problems labeled as "Other", we could not find appropriate formalization in any local theories. For instance, one of them asks to find the shortest path on the side-surface of a cone that connects the two ends of a diameter of the bottom face. We here should note that current categorization is based on informal observations on human problem-solving procedures. Some of the problems may require stronger language and axioms to fulfill the true end-to-end solution, and hence may be out of reach of the current formal reasoning technologies. This is not very surprising considering such a "basic fact" as Jordan Curve theorem ("a simple closed curve in the 2D plane divides its complement into two regions: inside and outside") has only been proved as late as in the 20th century (Veblen 1905).

Solving RCF Problems

As shown above, 40% of the problems (those categorized in "ZF set theory") are apparently not solvable in our current system. We also found nine out of the ten problems categorized as "PA" in Table 1 could not be solved with our current implementation based on a prover in Mathematica 9.0 even when we translate the problems into first-order formulas as simply as possible. We thus focus here on the 47 problems categorized as "RCF" problems to have an estimate on the practicality of solving them in a true end-to-end setting.

Table 2: Results of Solving RCF Problems

	Translation	
	Manual	Semi-Auto
Solved	28	23
Time Out	12	14
Impl. Issue	4	2
No RCF Formula	2	2
Unknown Directive	1	1
Translation Error	None	3
Grammar Design	None	2
total	47	47

To diagnose the failures more closely, we tested the system in two settings. In the first setting, we manually translated the 47 "RCF" problems to our semantic language. The translation was done as faithfully as possible to the textual expressions of the problems. In the second setting, we derived the logical translations of the problems from their linguistically annotated text. The annotations (for the 47 problems) included 865 dependency edges for 111 sentences, in which 81% of the edges were directly taken from the output of a dependency parser and the rest were by manual corrections, 29 paraphrases of coreferential phrases, 360 semantic annotations for math expressions, and 85 inter-sentence logical connectives. Two-thirds of the inter-sentence logical connectives were conjunctions ($\&$; see §2) and the rest were implications ($\Rightarrow$; see §2). In the course of the semi-automatic translation, we found most problems included a few words (or unknown usages of known words) not in our current CCG lexicon. We added 67 lexical entries in total to cover such words, most of which were different expressions for the concepts already in the semantic vocabulary. The experimental results hence should be interpreted as an upper-bound disregarding the out-of-vocabulary problem.

We classified the results on the 47 problems as in Table 2. The first row (labeled "Solved") shows the number of the problems for which the system produced correct answers (i.e., the correct values or conditions for "Find" directives and a proof by QE for "Show" directives). The system thus solved roughly 60% of the problems on the manual translations and 50% on the semi-automatic translations.

The second row ("Time Out") shows the number of problems on which the computation time exceeded the predetermined threshold (15 min). We are quite happy with the small difference between the numbers of time-out on manual and semi-auto translation. The semi-auto translation often produces massive formula as a result of mechanical, strictly word-by-word translation. The ability of the QE solver in handling such huge formulas gives much freedom in pursing principled analysis of natural language semantics without worrying much about the computational cost in the solver.

Table 3: Results on Univ. Tokyo Mock Test

Science Course (full score: 120)			
Test set	Translation		Human
	Manual	Semi-Auto	Avg.
2013 Jul	40	40	21.8
2012 Nov	40	40	32.5
2012 Jul	25	18	29.8

Humanities Course (full score: 80)			
Test set	Translation		Human
	Manual	Semi-Auto	Avg.
2013 Jul	40	40	24.9
2012 Nov	20	8	25.7
2012 Jul	40	40	30.9

The third row ("Impl. Issue") shows the number of the problems on which the system failed to solve due to several technical issues in interfacing with the CAS systems. The sum of the problems in the first three rows (Solved+ Time Our + Impl. Issue) are those for which our simple rewriting algorithm found a formula expressible in RCF. There were only two problems on which the algorithm failed (both on the manual and the semi-auto translations) to find such a formula ("No RCF Formula"). In one of them, we need to reconstruct the shape (i.e., defining inequalities) of an infinitely long triangular prism only by knowing the shapes of its cut sections with a moving plane. The reasoning is not very difficult but in the current framework we cannot handle it unless introducing many ad-hoc rewriting rules.

One problem ("Unknown Directive") could not be handled because it asks for a natural language description of a shape (ellipse) rather than its defining equation. For three problems, the semi-automatic translation produced wrong formulas ("Translation Error"). This was caused because the CCG derivation trees produced by the two-step parsing may include both wrong and correct translations due to the ambiguity of the grammar. We currently use a simple ranking function based on the shape of derivation trees to select a translation. The failures indicate that in some cases we need to utilize the solver to select a correct translation; wrong translations often give non-sensical answers, such as "there is no answer", but we can know it only by solving. Finally, in two problems we found grammatical constructions that cannot be properly analyzed within the current basic design of our CCG grammar ("Grammar Design").

Experiments on Tokyo Univ. Entrance Examination Mock Tests

We evaluated the prototype system on the University of Tokyo entrance exam mock tests held by one of the largest cram schools in Japan (Yoyogi Seminar). The problems in the mock tests are roughly at the same difficulty level as the real entrance exams. There are two types of tests, one is for future applicants for the Univ. of Tokyo science courses and the other is for humanities courses. Both types of the mock tests are sat by thousands of test takers.

The experimental setting is the same as before: we run the system both on manual translation of the problems and those semi-automatically derived from annotated texts. Table 3 shows the results on the six latest test sets that were available to us at the time of writing. The successfully solved problems included those on 2D and 3D geometry, linear algebra, calculus and natural number arithmetic.

On the semi-automatic setting, the system failed to solve two problems that were solved on their manual translations. One of the failures was due to a wrong translation of "two" in a problem that starts by saying "You have two functions: $f(x) = ax + b$ and $g(x) = x + 1$." From several conditions given later in the problem, we eventually find $a = 1$ and $b = 1$; hence the functions f and g are actually identical; i.e., there is only *one* function. Meanwhile, on another problem, the RCF-QE solver failed to process both the manually and semi-automatically translated inputs within an allotted time. Later inspection revealed that a modest estimate of the necessary computation time is far beyond the capacity of any conceivable hardware: in the course of the computation, a decomposition of $\mathbb{R}^7$ into more than 10^{51} cells have to be considered. It clearly indicates the need for further investigation on the problem representations to proceed much far in the direction envisaged in the current paper.

Overall, the results are encouraging. In four out of the six test sets, the scores attained by the prototype system matched the averages of human test takers. We should note again that the results are optimistic estimates of the performance of a true end-to-end system wherein the linguistic annotations on the text are automatically produced by language processing. The results however show that the overall architecture of our system is a reasonable choice at least as a first step toward a true end-to-end wide coverage problem solver.

Related Work

Since Bobrow's STUDENT (1964), there is a long line of AI work on solving math problems written in natural language (Mukherjee and Garain (2008) gives a recent survey). Most previous work however focused on developing an intermediate language specialized to a certain type of problems, such as arithmetic word problems, rather than covering various problems in a single formal system (e.g., ZF set theory) as in our approach. Meanwhile, there have been several proposals and attempts to equip proof verifiers and provers with (controlled) natural language interfaces (Ganesalingam 2009; Cramer, Koepke, and Schröder 2011; Ranta 2011). To our knowledge, however, no previous work in that line have quantitatively evaluated such systems on real math text.

Conclusion

We have presented a prototype of a math problem solving system wherein a grammar-based translation of math problem text is combined with automated reasoner, through a equivalence-preserving formula rewriting. Empirical evaluation on real and mock university entrance exam problems showed positive results that indicate the viability of a true end-to-end problem solving system in the direction pointed in the current paper.

Acknowledgments

This research was supported by Todai Robot Project at National Institute of Informatics. We are gratefully acknowledge Gakko Hojin Takamiya Gakuen Yoyogi Seminar for the University of Tokyo entrance exam mock test data.

References

Bekki, D. 2010. *Nihongo-bunpou no keishiki-riron (in Japanese)*. Kuroshio Shuppan.

Bobrow, D. G. 1964. *Natural language input for a computer problem solving system*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Brown, C. W. 2003. QEPCAD B - a program for computing with semi-algebraic sets using CADs. *SIGSAM Bull.* 37(4):97–108.

Clark, S., and Curran, J. R. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Comput. Linguist.* 33(4):493–552.

Collins, G. E. 1975. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, May 20-23, 1975*, volume 33 of *Lecture Notes in Computer Science*, 134–183. Springer-Verlag.

Cramer, M.; Koepke, P.; and Schröder, B. 2011. Parsing and disambiguation of symbolic mathematics in the naproche system. In *Proceedings of the 18th Calculemus and 10th International Conference on Intelligent Computer Mathematics*, MKM'11, 180–195. Berlin, Heidelberg: Springer-Verlag.

Dolzmann, A., and Sturm, T. 1995. Simplification of quantifier-free formulas over ordered fields. *Journal of Symbolic Computation* 24:209–231.

Dolzmann, A., and Sturm, T. 1997. REDLOG: computer algebra meets computer logic. *SIGSAM Bull.* (2):2–9.

Ganesalingam, M. 2009. *The Language of Mathematics*. Ph.D. Dissertation, University of Cambridge.

Groenendijk, J., and Stokhof, M. 1991. Dynamic predicate logic. *Linguistics and Philosophy* 14(1):39–100.

Haghighi, A., and Klein, D. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 385–393. Los Angeles, California: Association for Computational Linguistics.

Hong, H. 1993. Quantifier elimination for formulas constrained by quadratic equations. In *Proceedings of the 1993 international symposium on Symbolic and algebraic computation*, ISSAC '93, 264–274. New York, NY, USA: ACM.

Iwane, H.; Yanami, H.; Anai, H.; and Yokoyama, K. 2013. An effective implementation of symbolic-numeric cylindrical algebraic decomposition for quantifier elimination. *Theor. Comput. Sci.* 479:43–69.

Iwane, H.; Higuchi, H.; and Anai, H. 2013. An effective implementation of a special quantifier elimination for a sign definite condition by logical formula simplification. In *CASC*, 194–208.

Kudo, T., and Matsumoto, Y. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, 1–7. Stroudsburg, PA, USA: Association for Computational Linguistics.

Lin, Z.; Kan, M.-Y.; and Ng, H. T. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, 343–351. Stroudsburg, PA, USA: Association for Computational Linguistics.

Loos, R., and Weispfenning, V. 1993. Applying linear quantifier elimination. *The Computer Journal* 36(5):450–462.

McCarthy, J. 1964. A tough nut for proof procedures. Ai memo, MIT.

Mukherjee, A., and Garain, U. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artif. Intell. Rev.* 29(2):93–122.

Ranta, A. 2011. Translating between language and logic: What is easy and what is difficult. In Bjørner, N., and Sofronie-Stokkermans, V., eds., *Automated Deduction – CADE-23*, volume 6803 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 5–25.

Steedman, M. 2001. *The Syntactic Process*. Bradford Books. Mit Press.

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

Uematsu, S.; Matsuzaki, T.; Hanaoka, H.; Miyao, Y.; and Mima, H. 2013. Integrating multiple dependency corpora for inducing wide-coverage japanese ccg resources. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1042–1051. Sofia, Bulgaria: Association for Computational Linguistics.

Veblen, O. 1905. Theory on plane curves in non-metrical analysis situs. *Transactions of the American Mathematical Society* 6(1):pp. 83–98.

Wilson, D. J.; Bradford, R. J.; and Davenport, J. H. 2012. Speeding up cylindrical algebraic decomposition by Gröbner bases. *CoRR* abs/1205.6285.