AiFu at SemEval-2019 Task 10: A Symbolic and Sub-symbolic Integrated System for SAT Math Question Answering

Keyu Ding[1,2,6], Yifan Liu[1,2], Yi Zhou[3,5], Binbin Deng[1,2], Chaoyang Peng[1,2], Dinglong Xue[1,2], Qinzhuo Wu[3], Qi Zhang[3] and Enhong Chen[5]

[1]iFLYTEK Research
[2]State Key Laboratory of Cognitive Intelligence, iFLYTEK, P.R. China
[3]Shanghai Research Center for Brain Science and Brain-Inspired Intelligence/ZhangJiang Lab
[4]School of Computer Science, Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
[5]School of Natural and Computational Sciences, Massey University
[6]University of Science and Technology of China
{*kyding,yfliu7,bbdeng,zypeng,dlxue*}*@iflytek.com*
yzhou@bsbii.cn
{*qzwu17,qz*}*@fudan.edu.cn*
cheneh@ustc.edu.cn

Abstract

AiFu has won the first place in the SemEval-2019 Task [10] - "Math Question Answering"(Hopkins et al.) competition. This paper is to describe how it works technically and to report and analyze some essential experimental results.

1 Introduction

Recently, math question answering has attracted a lot of attention in the AI community, both in academia and in industry (Matsuzaki et al., 2017) (Wang et al., 2017) (Huang et al., 2016) (Wang et al., 2018) (Hosseini et al., 2014)(Huang et al., 2018a) (Huang et al., 2018b) (Kushman et al., 2014) (Liang et al., 2017) (Mitra and Baral, 2016) (Zhou et al., 2015) (Roy and Roth, 2017)(Hopkins et al., 2017). On one side, it raises a difficult yet workable challenge for the current development of AI research. In order to tackle this challenge, one has to integrate and advance many subareas in AI including knowledge representation and reasoning, machine learning, natural language understanding and image understanding. On the other side, math question answering itself has important commercial value in the AI+Education industry.

Against this backdrop, SemEval-2019 organizes a competition on math question answering, namely Task 10 (Hopkins et al.). In this task, an opportunity is provided for Math Question-Answering systems to test themselves on a benchmark that consists of many math questions collected from the US Math Scholastic Achievement Test (SAT).

We have implemented a prototype system, called AiFu, to tackle this challenge, and it has won the first place at the end. AiFu is an integrated system that combines the state-of-the-art approaches from many important subareas in AI, and more importantly, it also develops and justifies some new ideas and techniques.

This paper is to describe how AiFu works technically and to report and analyze some essential experimental results. In the next section, we go through the technical details of AiFu that consists of many essential components including representation, reasoning and natural language understanding. In Section 3, we report our experimental results and shed new insights on why AiFu works in some cases but not in others. Finally, we conclude this paper and point out some future directions.

2 Method

Figure 1 depicts the overall architecture of AiFu. Given a mathematical question, a translator is used to convert it into its internal representation, which is sent to an encoder to further convert it into a math representation that can be directly used by

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 900–906
Minneapolis, Minnesota, USA, June 6–7, 2019.

the SMT solver Z3 (De Moura and Bjørner, 2008). Finally, by calling Z3, the solution of the original mathematical question is obtained.

2.1 Internal Representation

We use an internal representation language, called Verb Connection Formula (VCF), to bridge the gap between mathematical questions and their formal mathematical counterparts.

VCF is based on assertional logic (Zhou, 2017), in which all mathematical objects are formalized as either individuals (constants and variables), or concepts, or operators (functions and relations). For instance, the natural language sentence "the integer x equals to 3" is transformed to "Integer(x), Equal(x,3)" in VCF, where "x" and "3" are individuals, "Integer" is a concept and "Equal" is a Boolean operator, i.e., relation. Meta level mathematical concepts such as equation and inequality are represented as concepts in VCF too. For instance, an equation $9 + 3^{n+2} = m$ in the question is transformed to "Equation(9+3**(n+2)=m)" in VCF, and an inequality $xyz \neq 0$ is transformed into "Inequality(x*y*z!=0)". VCF uses the symbol $:-$ for representing the implication relationship between statements. For instance, the VCF representation of the natural language sentence "When n is a positive integer, $9 + 3^{n+2} = m$" is "(Equation(9+3**(n+2)=m)):-(Positive(n),Integer(n))".

2.2 Translator

The translator transforms mathematical questions to corresponding statements in VCF.

Segmentation and POS tagging: Our translator uses the Stanford NLP parser (Chen and Manning, 2014) for segmentation and POS tagging. In order to handle Math questions, we introduce two new POS taggers, namely "FORM" for indicating Math formula and "VAL" for indicating variable. For example, the result of POS tagging of "When n is a positive integer, $9 + 3^{n+2} = m$" is "When/WRB n/VAL is/VBZ a/DT positive/JJ integer/NN ,/PUNCTUATION $9 + 3^{n+2} = m$/FORM".

Semantic parsing:

We adopt a top-down rule-based template approach for semantic parsing, i.e., translating math questions in English to statements in VCF. As shown in Table 1, we consider five basic clause types, corresponding to five syntactic structures respectively.

Type	Description	Examples
LEAF	a single word	"If","percent", "of"
NOUN	entity with adjunct words	"125 percent"
PREP	relation with multiple entities	125 percent of x
PRED	clause with operators	"125 percent of x is 150"
CONJ	causal connection between two clauses	"if 125 percent of x is 150,what is x percent of 75"

Table 1: Clause types

Algorithm 1 illustrates how to construct the semantic parsing tree for each sentence in the question from top to down. The root must be the sentence itself. Each node is assigned with one of the five types according to hand-crafted rule-based templates. Based on which, we decompose it into several child nodes correspondingly as different clause types result in different kinds of decomposition according to the templates. Note that each leaf node must be assigned with LEAF.

Algorithm 1: Semantic parsing algorithm

```
   input : a sentence in the question
   output: a list of VCF statements
1  root = original sentence;
2  Stack = Empty;
3  VCF_Stack = Empty;
4  Stack.push(root);
5  while Stack is not Empty do
6      current_node = Stack.pop();
7      templates = get_valid_templates(current_node);
8      template = choose_template(templates);
9      current_node.VCF_template = get_VCF_template(template);
10     VCF_Stack.push(current_node);
11     nodes = get_child_nodes(template,current_node);
12     for node in nodes do
13         current_node.child_nodes.add(node);
14         if type(node) != LEAF then
15             Stack.push(node);
16         end
17     end
18 end
19 while VCF_Stack is not Empty do
20     current_node = VCF_Stack.pop();
21     current_node.VCF=get_VCF_from_template_and_child_node();
22 end
23 return root.VCF
```

Figure 2 illustrates a semantic parsing tree example, in which each node is associated with one of the five basic types. According to this semantic parsing tree, we compute the resulting VCF statements from bottom to up recursively. For instance, the node "125 percent/NOUN" is converted into "NumberPer-

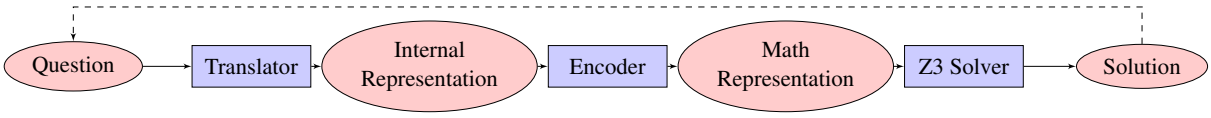

Figure 1: The architecture of AiFu

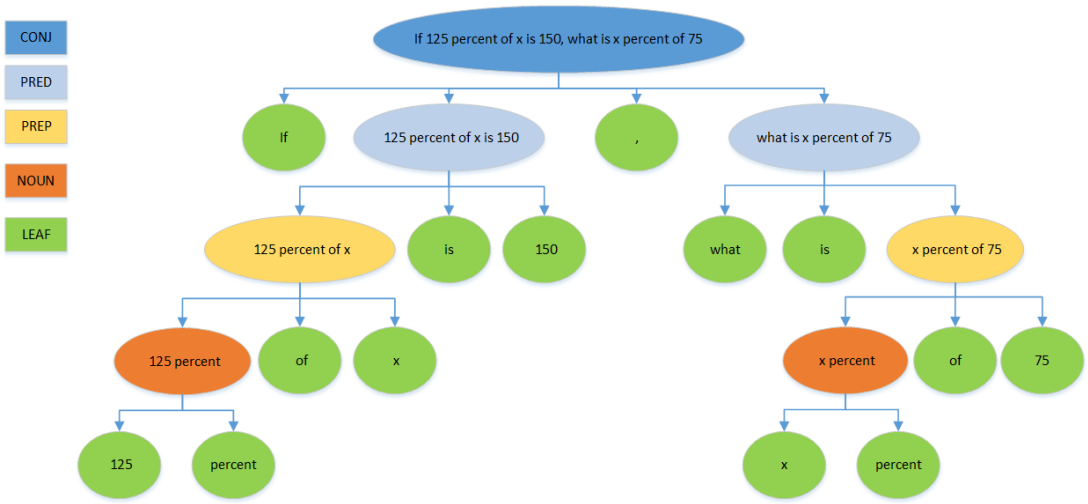

Figure 2: Semantic parsing tree: a case study

cent(125)", while the node "What is x percent of 75/PRED" is converted into "NumberPercent(x), Of(75,x,rs_a), Be(rs_b,rs_a), What(rs_b)". Finally, the root node, i.e., the original sentence, is converted into "(NumberPercent(x), Of(75,x,rs_a), Be(rs_b,rs_a), What(rs_b)) :- (NumberPercent(125), Of(x,125,rs_c), Be(rs_c,150))".

2.3 Encoder

The encoder further converts statements in VCF to formulas that can be accepted by Z3. Again, we use a rule based template approach for this purpose. There are two types of encoding templates. One is used to unify all different kinds of concepts/operators in VCF to a set of predefined concepts/operators that are accepted by Z3. For instance, a VCF statement "add(3,5,x)" is normalized as "Equal(3+5,x)". While the operator "add" is not a Z3 acceptable one, "Equal" and "+" are. Another type of template is to unify new entities that are created by VCF. For instance, the sentence "x is an integer" is converted to VCF statements "Be(rs_a,x),Integer(rs_a)" in the translator, which is further encoded as "Integer(x)" in the encoder by unifying the two entities "x" and "rs_a". At first glance, it seems tedious to introduce extra entities in the translator. However, this is exactly the reason why we need an intermediate representation language VCF because machines cannot directly understand what "x is an integer" really means.

By using the encoding templates, the VCF statements obtained in Figure 2 is converted into "Var: x:Real, Equations: x*125% = 150, Target: x%*75", which can be directly sent to the Z3 solver.

A large portion of SAT math questions can be done in this way, thus are suitable to use Z3 as the solver. Nevertheless, for some mathematical concepts such as progression, set, list and odd/even numbers, we need to make extra effort to formalize them in the modulo theory linear arithmetic. For instance, Odd(x) ("x is an odd number") can be converted into Equation($x\%2 = 1$). While the former cannot be directly encoded in linear arithmetic, the latter can.

2.4 Z3 Solver

Similar to some previous approaches (Hopkins et al., 2017), we also call the Z3 solver[1] as our reasoning engine. Z3 is a widely used Satisfiability Modulo Theories (SMT) solver, for solving problems that are represented in classical logic augmented with modulo theories, e.g., linear arithmetic.

However, Z3 has inherited difficulties on solving nonlinear equations, e.g., $3\sqrt{x} - 7 = 20$. In

[1]https://github.com/Z3Prover/z3

this case, we use SymPy [2] (Meurer et al., 2017), a Python library for symbolic mathematics, as the backup.

2.5 More on Geometry and Open Categories

For answering geometry questions, one has to understand diagrams. For this purpose, we first use the Optical Character Recognition (OCR) tool pytesseract[3] to obtain character information. Then, we follow the combined text and diagram understanding approach GeoS (Seo et al., 2015).

Understanding open-vocabulary algebra questions is a critical challenge. At first glance, it seems that the SAT open-vocabulary algebra subdataset is quite similar to Math23k (Wang et al., 2017). Hence, we attempted to use a Seq2Seq approach (Wang et al., 2017) that transforms math questions directly to their corresponding mathematical meanings. However, this attempt was not successful, mainly because of the following two reasons. First, questions in Math23k are much simpler. Second, Math23k is much larger in terms of volume.

Hence, we shifted back to a rule-based approach. We first use SVM to classify the type of questions. Based on which, regular expressions are used to transform questions in natural languages to statements in VCF, similar to that for the closed category described above. It turns out that its performance is slightly better than the Seq2Seq approach, yet still far from satisfactory.

2.6 Sub-symbolic System

We call the framework (see Figure 1) described above the "symbolic system" as it mainly uses a symbolic approach. However, some math questions remain unsolved. Hence, we also implement a guess system as a complementary counterpart. A simple guess system would be just a random guesser. Nevertheless, in `AiFu`, we use a neural-network based sub-symbolic approach, called the "sub-symbolic system" instead based on sentence embedding.

We treat the math question answering problem as a classification problem. We combine the question as well as a candidate choice into one sentence, and use a sentence embedding approach InferSent (Conneau et al., 2017) to embed it into a vector of 4096 dimensions. Then, we construct a simple four-layer fully-connected feed-forward neural network, in which the input is the 4096-dimension sentence embedding, the output is the 5 classes of answers and the two hidden layers both contain 1024 nodes. Finally, we train the network with the training dataset provided by the organizer.

3 Results

Table 2 reports the overall final results of `AiFu` on the test dataset. "Symb" is the symbolic system described from Sections 2.2 to 2.5, "Sub-S" is the sub-symbolic system described in Section 2.6, and "Integ" is the integrated system `AiFu` that combines them both by answering those questions not answered by the symbolic system with the sub-symbolic system. While "Acc" refers to the standard accuracy, "Pena Acc" refers to the penalized accuracy by deducting 0.25 points each for a wrong answer.

All in all, `AiFu` achieves an overall accuracy of 45% as well as an overall penalized accuracy of 36%. In particular, on the closed-vocabulary algebra category, it achieves a relatively high accuracy 70% (66% for penalized accuracy in contrast). The symbolic system plays a more critical role as it alone achieves 63% on answering closed-vocabulary algebra questions. More importantly, the symbolic system has a very high precision of 96%, thus has almost no penalization on all categories. In addition, unlike the sub-symbolic system, the symbolic system is fully explainable. However, it can be observed that all of these systems still perform poor on the Geometry and the open categories, especially when measured by penalized accuracy.

Table 3 illustrates the semantic parsing accuracy rate on closed-vocabulary algebra questions. It can be observed that, on the training and development datasets, we can achieve a relatively high accuracy of 84%. Nevertheless, it drops down to 71% on the test dataset. This is mainly because of the generalizability issue of templates.

In order to further analyze the genrealizability issue of templates, we consider the effects on the number of templates. Figure 3 shows how the template number affects the semantic parsing accuracy. It can be observed that, on the training dataset, the accuracy rates rapidly goes up to 50 % by the first 400 templates, then reaches 70% by 900 templates. Then, it slowly climbs up to 80% with 500 templates more. However, the growth

[2]https://www.sympy.org/en/index.html
[3]https://pypi.org/project/pytesseract/

Version	Acc (overall)	Acc (closed)	Acc (geo)	Acc (open)	Pena Acc (overall)	Pena Acc (closed)	Pena Acc (geo)	Pena Acc (open)
Symb	0.29	0.63	0.08	0.03	0.29	0.62	0.07	0.03
Sub-S	0.23	0.22	0.20	0.25	0.11	0.10	0.09	0.14
Integ	**0.45**	**0.70**	**0.26**	**0.26**	**0.36**	**0.66**	**0.14**	**0.16**

Table 2: Overall Results

Dataset	Closed
Training	84% (711/846)
Development	84% (187/222)
Test	71% (326/459)

Table 3: Semantic parsing accuracy

Error Analysis	Training
Non-linear question	19%
Contain math concept	49%
Contain definition	9%
Other tricky question	23%

Table 4: Questions that cannot be solved by Z3

rate drops down dramatically afterwards, making it very difficult to improve. The main reason is that SAT math questions have many long-tail questions that cannot be covered by ordinary templates.

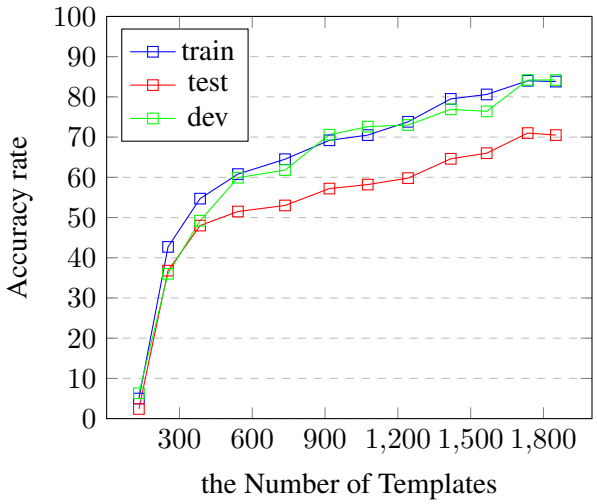

Figure 3: Effects on the number of templates

There are some questions that can be correctly parsed by our translator and encoder but failed to be solved by Z3. Among all, nearly 12% (100 out of 846) belong to this case in the training dataset. We further analyze their features. Among them, 49% need to use new math concepts, e.g., prime number, that cannot be represented in Z3. Around 19% of the questions are non-linear equations, which are very difficult for Z3. Finally, there are 9% of the questions need to define new objects, and the rest 23% are other kinds of tricky questions.

4 Conclusion

In this paper, we presents `AiFu`, a system that has won the first place in the SemEval-19 "Math Question Answering" competition. `AiFu` is a combined system that enhances a symbolic system with a sub-symbolic guesser. In `AiFu`, the symbolic system plays the most vital role, which itself can achieve a relatively high accuracy with many merits on closed-vocabulary algebra questions. Many state-of-the-art approaches are used and integrated in `AiFu`. Some new techniques are proposed including our new internal representation language VCF and our template structures.

For future work, the most important task is to improve the translator, which is the bottleneck, especially on geometry and open-vocabulary algebra questions. We believe that new foundations are needed, possibly requiring a deep integration of symbolic and sub-symbolic approaches.

References

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *Proceedings*

of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'08/ETAPS'08, pages 337–340, Berlin, Heidelberg. Springer-Verlag.

Mark Hopkins, Ronan Le Bras, Cristian Petrescu-Prahova, Gabriel Stanovsky, Hannaneh Hajishirzi, and Rik Koncel-Kedziorski. Semeval-2019 task 10: Math question answering.

Mark Hopkins, Cristian Petrescu-Prahova, Roie Levin, Ronan Le Bras, Alvaro Herrasti, and Vidur Joshi. 2017. Beyond sentential semantic parsing: Tackling the math sat with a cascade of tree transducers. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 795–804. Association for Computational Linguistics.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 523–533.

Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. 2018a. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 213–223.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Danqing Huang, Jin-Ge Yao, Chin-Yew Lin, Qingyu Zhou, and Jian Yin. 2018b. Using intermediate representations to solve math word problems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 419–428.

Nate Kushman, Luke Zettlemoyer, Regina Barzilay, and Yoav Artzi. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 271–281.

Chao-Chun Liang, Yu-Shiang Wong, Yi-Chung Lin, and Keh-Yih Su. 2017. A goal-oriented meaning-based statistical multi-step math word problem solver with understanding, reasoning and explanation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 5235–5237.

Takuya Matsuzaki, Takumi Ito, Hidenao Iwane, Hirokazu Anai, and Noriko H. Arai. 2017. Semantic parsing of pre-university math problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 2131–2141.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. 2017. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3082–3088.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *EMNLP*.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5545–5552.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 817–822.

Yi Zhou. 2017. From first-order logic to assertional logic. In *Artificial General Intelligence - 10th International Conference, AGI 2017, Melbourne, VIC, Australia, August 15-18, 2017, Proceedings*, pages 87–97.