# A Sequence-to-Graph Model for Visualizing Math Operation Programs

**Avi Bleiweiss**
BShalem Research
Sunnyvale, CA, USA
avibleiweiss@bshalem.onmicrosoft.com

## Abstract

We propose to improve student intuition for interpreting solutions of math word problems. Rather than a linear textual sequence that embeds operations and operands, often in an unclear manner, we offer a succinct program visualization of annotated formulas.

## 1 Description

In Table 1, we review our contribution to visualize an annotated formula in red.

**problem:** a horse is tethered to one corner of a rectangular grassy field 36 m by 20 m with a rope 18 m long . over how much area of the field can it graze ?

**formula:** divide(multiply(power(18, const_2), const_pi), const_4)

**edge list:** (divide, multiply), (multiply, power), (power, 18), (power, const_2), (multiply, const_pi), (divide, const_4)

**program plot:**

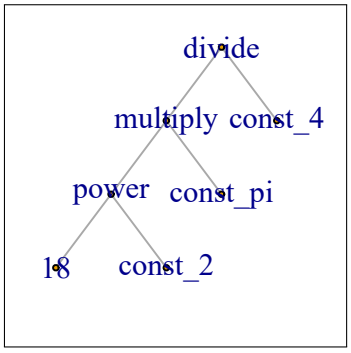


Table 1: MathQA generates an annotated formula for each math word problem, of which we construct a graph and extract an edge list. The program plot uses symbolic labels that abide by the MathQA lists of operations and constants. Graph plot generated by iGraph.

In our experiments, we used MathQA (Amini et al., 2019), a large-scale dataset composed of GRE-level math problems that is widely used as a benchmark for university admission in the US. MathQA introduces a representation language to model operation programs in both a linear and annotated forms, and explores a neural architecture that exploits a sequence-to-program model and integrates knowledge of problem classification.

MathQA has the pivotal objective for an operation program to yield the correct solution once all operations are executed. While our study performs a sequence-to-graph transform and highlights the importance of formula visualization to enhance student learning experience and comprehension of math problems.

We note that the graph complexity of a programmatic math problem is fairly low with about fifteen nodes on average, and hence the clarity provided by graph rendering. In our experiments, we used iGraph (Csardi and Nepusz, 2006) for network modeling and interpretation. iGraph requires that the names of program operations and literals must be explicitly unique, and programs that mix unary and binary operations are supported to a limited extent.

## References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Gabor Csardi and Tamas Nepusz. 2006. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.