A Computational Theory of Inference for Arithmetic Explanation

Albert Goldfain

Department of Computer Science and Engineering
Center for Cognitive Science
University at Buffalo
Buffalo, NY 14260
ag33@cse.buffalo.edu

Abstract

Mathematical understanding can be measured by a cognitive agent's ability to explain itself, i.e., answer relevant questions about its mathematical activities. Two inference techniques, rule-based inference and path-based inference, are applied to an implemented computational cognitive agent using the SNePS knowledge-representation, reasoning, and acting system.

1 Introduction

When engaged in classroom mathematical activities, students are often encouraged to "show their work" as they progress towards a solution and to "explain their answers" after a solution has been found. This justification, required well before a student learns how to produce rigorous logical proofs, is a demonstration that the student understands *how* the problem is solved and *why* the obtained result is a solution. Students "show their work" by explicitly providing the intermediate computations needed to reach a solution. When confronted with a problem of averaging, e.g., 2, 3, 15, and 20, students should show the intermediate sum 2+3+15+20=40 and the intermediate division $40/4 = 10$. Students show that they understand averaging *in terms of* the simpler operations of addition and division.

Unlike human students, computers and calculators are usually expected to produce fast, reliable results *without* an explanation. These results often take the form of a numerical output devoid of intermediate calculations, linguistic content, and problem-specific semantic information. In such a role, the computer is a tool for the human user, a tool that produces the correct answers without justification. However, in the field of artificial intelligence, there are several reasons to design computational agents that can both produce the correct answers and can explain their answers. Such agents could better

communicate and interoperate with human users. Knowing the "how" and "why" behind mathematical routines is required for a computational model of mathematical cognition in a cognitive agent. Extending these abilities can be a great benefit to autonomous embodied agents that must reason in the real world with minimal human interaction.

This paper is a preliminary investigation of the inferential and representational requirements for a computational agent that can produce mathematical explanations at the level of basic arithmetic and counting. Explanation is treated as a justification of procedural action (the kind of solution a student might give), rather than as a formal proof (the kind of solution a mathematician might give).

2 Question Answering

To probe a cognitive agent's mathematical understanding, a series of questions can be posed to the agent after it has completed a mathematical activity. For example, if an agent has just determined that 2 is the greatest common divisor of 8 and 6, a highly idealized explanation dialogue might look like the following:

Q1: Why is 2 the greatest common divisor of 8 and 6?

A1: 2 is the greatest of the common divisors of 8 and 6.

Q2: Why is 2 a common divisor of 8 and 6?

A2: 2 is a divisor of 8 and 2 is a divisor of 6.

Q3: Why is 2 a divisor of 6?

A3: There is a number that, when multiplied by 2, gives 6, and that number is 3.

Q4: Why is 2 times 3 = 6?

A4: Multiplication is repeated addition; 2 plus 2 is 4 and 4 plus 2 is 6

Q5: Why is 2 plus 2 = 4?

A5: When I count from 2 for two numbers I end up at 4

Q6: How do you know that you will end up at 4?

A6: I counted two groups of oranges, with 2 oranges in each, ending up with 4 total oranges.

Q7: What is 2?

A7: It is a number and the greatest common divisor of 8 and 6.

Q8: What is a number?

A8: Some examples are 2,4,6 and 8 . . . It is something that can be counted, added, multiplied . . . and something that can be the result of finding a greatest common divisor

A human addressee will likely stop short of such a detailed answer. However, for computational agents, such a dialogue will be a useful Turing-test-style demonstration of mathematical understanding. The ability to produce such answers demands a representational and inferential capacity beyond that of just finding a greatest common divisor. During a dialogue, an agent may need to: (1) call upon linguistic information (e.g., for A2 above), (2) call upon real-world experiences (e.g., for A6 above), (3) infer the relationships between

procedures and their constituent sub-procedures (e.g., for A4 above), and (4) infer relationships between concepts (e.g., for A8 above). The question for a computational agent then becomes: where does this (required) additional knowledge come from?

One answer to this question is to design an agent that accumulates knowledge *during* a mathematical activity. The ability to assimilate newly inferred material with existing knowledge is essential for mathematical understanding [2]. Mathematical understanding is driven by *doing* things, not simply by *thinking* about things. This is consistent with the cognitive foundations of mathematics presented by Lakoff & Núñez [4], in which a metaphoric relationship is established between a human activity (such as object collection) and a formal operation (such as addition).

The activity-driven nature of mathematical understanding can be witnessed even in the developmentally early routine of counting. Children turn a routine that is nothing more than a meaningless recitation of ordered words (much like a nursery rhyme) into a tool for ascribing cardinal size and ordinal position to real-world entities. The semantics of counting routines arises from children performing tasks that require counting, not from the contemplation of number-name meanings. Outside the context of the number line, number-names are meaningless identifiers.

3 SNePS

My computational theory is implemented in the SNePS knowledge-representation, reasoning, and acting system [11]. The fundamental data-structure of SNePS is a propositional semantic network. A SNePS network represents the beliefs of Cassie, the SNePS cognitive agent. A semantic network is abstract enough to represent both numeric and linguistic information. This is one of the reasons semantic networks have been used in models of arithmetic word-problem solving [1]. Numeric information is basically syntactic [8], but obtains a conceptual-role semantics [6] through integration with a system such as SNePS.

Cassie's acting system is SNeRE (the **SNe**PS **R**ational **E**ngine [3]). Acts in SNePS are either primitive or complex. Primitive acts are the fundamental repertoire of acts available to Cassie. Complex acts are composed of sets of primitive acts that are structured by control acts (i.e., acts for sequencing, iteration, and conditionals) and are bound to plans for performing them with the `ActPlan` predicate function. SNePS has a uniform representation for both conceptual and procedural knowledge so that all of Cassie's SNeRE plans are stored in the same network as her conceptual knowledge. Most importantly, Cassie can add beliefs to her network *while* she is acting, constituting an episodic memory of the act, which can be accessed during an act to prompt further action or after an act for answering questions.

SNIP, the **SN**ePS **I**nference **P**ackage, enables inferential operations over

the beliefs in Cassie's network. This allows SNePS to serve as a logical rule-based system. Commands for finding and deducing nodes (representing propositions) allow the user to ask Cassie questions and are the foundation for the path-based and rule-based inference techniques described in the next section.

3.1 Rule-Based Inference

At the highest level of abstraction, an `ActPlan` for an arithmetic act is set up as follows:

```
all(x,y)({Number(x),Number(y)} &=> ActPlan(Add(x,y),CountAdd(x,y))).
```

The semantics of such a statement is roughly: If you can infer that both x and y are numbers, then a plan for doing this generic arithmetic operation (e.g., `Add`) is this specific arithmetic operation (e.g., `CountAdd`). When Cassie is told to perform one of the generic arithmetic acts on two given inputs, say `perform Add(2,3)`, she produces the following sequence of inferences: [1]

(i) Try to deduce a plan for the act `Add(2,3)`. This amounts to asking the open question `ActPlan(Add(2,3),?x)?`.

(ii) Once Cassie finds an asserted node corresponding to `Add(x,y)`, she backchains and wonders whether `Number(2)` and `Number(3)` are asserted beliefs.

(iii) Once Cassie determines that `Number(2)` and `Number(3)` both hold, she retrieves the plan corresponding to the act `Add(2,3)`. In this case, she finds `CountAdd(2,3)`.

(iv) Cassie then attempts to `perform CountAdd(2,3)` by finding a plan for it.

This cycle of actions continues until Cassie completes the task. When a primitive action is reached, Cassie does not try to further decompose the action.

During the course of executing a specific arithmetic operation, Cassie needs to store the result using the `CountSum` case frame. We also want to be able to say that each of these specific arithmetic results is a result for the abstract act. Cassie is given the following rule:

```
all(x,y,z)(CountSum(x,y,z) => Sum(x,y,z)).
```

This tells Cassie that a `CountSum` is a `Sum` *per se*. Thus, the moment Cassie believes the result `CountSum(2,3,5)`, she will infer `Sum(2,3,5)` by forward chaining. This provides a specific and generic access point for arithmetic results.

These arithmetic acts force Cassie to perform successively simpler operations. Rule-based inference establishes a connection between three things: a general act (e.g., addition), a specific plan for performing that act (e.g., count-addition), and a specific performance of that plan (e.g., the count-addition of

[1] Cassie also checks whether an act has any preconditions and any effects. Since the arithmetic operations involve mental acts only, we omit these inferences.

2 and 3). Since Cassie explicitly deduces propositions at the knowledge level using this technique, rule-based inference can be seen as a model of "conscious" inference (i.e., Cassie is attending to the information that triggers the inference).

3.2 Path-Based Inference

Paths in SNePS networks are ordered sequences of arc-labels between nodes. Paths are specified using a syntax similar to regular expressions [9]. Path-based techniques can be used as a model of unconscious inference for SNePS agents [10]. A relation between nodes in a given network may be inferred by the presence of a certain path between those nodes. This is an "unconscious" activity, because the newly inferred relation is added to the agent's belief space without an explicit knowledge level deduction.

For each new number generated by a counting procedure, Cassie builds a `Successor` relation to hold between that number and its immediate predecessor. Using path-based inference, this link can be exploited to generate the `GreaterThan` relation, which holds between each new number generated and *all* preceding numbers.

Path-based inference takes advantage of the fact that Cassie's knowledge is stored in a semantic network, in which a node's "meaning" is determined by its position in the network relative to other nodes [5,6]. This type of inference can also be used in SNePS to determine the conceptual-role semantics for vague concepts such as "number". To define a concept such as "number" in the context of a completed arithmetic activity, a series of path-based inferences can be performed to give the elements of the class number, to give the acts with argument type number, and to give the operations that have thus far resulted in numbers.

This technique can also be used to provide semantic meaning for arithmetic procedures. The possibility of treating an arithmetic operation as either a concrete object (as a node) and a process (as an `ActPlan`) during inference is an important result of using SNePS to model mathematical understanding (see [7]).

4 Conclusion and Further Research

Inference is central to explanations of all sorts. We have seen that for a domain such as arithmetic, in which complex procedures can be rigidly built up from simpler procedures, the knowledge inferred during an agent's action can provide a justification for that action. For a computational agent, the choice of a knowledge-representation system determines the methods of inference that can be applied. SNePS is particularly useful for implementing my theory because it can be used as a logical rule-based system (by applying node-based inference) or as a traditional semantic network (by applying path-based

inference). The agent implementation is still in its early stages and there are several avenues to pursue. [2]

References

[1] Greeno, J. G., *Instructional Representations Based on Research about Understanding*, in: *Cognitive Science and Mathematics Education*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1987 pp. 61–88.

[2] Hiebert, J. and P. Lefevre, *Conceptual and Procedural Knowledge in Mathematics: An Introductory Analysis*, in: *Conceptual and Procedural Knowledge: The Case of Mathematics*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986 pp. 1–27.

[3] Kumar, D., *From Beliefs and Goals to Intentions and Actions: An Amalgamated Model of Inference and Acting*, Technical Report 94-04, Department of Computer Science, State University of New York at Buffalo (1994).

[4] Lakoff, G. and R. Núñez, "Where Mathematics Comes From: How the Embodied Mind Brings Mathematics Into Being," Basic Books, New York, NY, 2000.

[5] Quillian, M. R., *Semantic Memory*, in: *Semantic Information Processing*, MIT Press, Cambridge, MA, 1968 pp. 216–270.

[6] Rapaport, W. J., *Holism, Conceptual-Role Semantics, and Syntactic Semantics*, Minds and Machines **12** (2002), pp. 3–59.

[7] Sfard, A., *On The Dual Nature of Mathematical Conceptions: Reflections on Processes and Objects as Different Sides of the Same Coin*, Educational Studies in Mathematics **22** (1991), pp. 1–36.

[8] Shapiro, S. C., *Representing numbers in semantic networks: prolegomena*, Proceedings of the 5th International Joint Conference on Artificial Intelligence (1977), p. 284.

[9] Shapiro, S. C., *Path-based and node-based inference in semantic networks*, ACM, New York, NY, 1978 pp. 219–225.

[10] Shapiro, S. C., *Cables, paths and "subconscious" reasoning in propositional semantic networks*, in: J. F. Sowa, editor, *Principles of Semantic Networks*, Morgan Kaufmann, San Mateo, CA, 1991 pp. 137–156.

[11] Shapiro, S. C. and W. J. Rapaport, *The SNePS Family*, Computers and Mathematics with Applications **23** (1992), pp. 243–275.

[2] I would like to thank William J. Rapaport and Stuart C. Shapiro for reading earlier drafts of this paper.