Modelling the Way Mathematics Is Actually Done

Joseph Corneli
University of Edinburgh
joseph.corneli@ed.ac.uk

Ursula Martin
University of Oxford
Ursula.Martin@cs.ox.ac.uk

Dave Murray-Rust
University of Edinburgh
d.murray-rust@ed.ac.uk

Alison Pease
University of Dundee
a.pease@dundee.ac.uk

Raymond Puzio
PlanetMath.org, Ltd.
rspuzio@planetmath.info

Gabriela Rino Nesin
University of Brighton
G.RinoNesin@brighton.ac.uk

Abstract

Whereas formal mathematical theories are well studied, computers cannot yet adequately represent and reason about mathematical dialogues and other informal texts. To address this gap, we have developed a representation and reasoning strategy that draws on contemporary argumentation theory and classic AI techniques for representing and querying narratives and dialogues. In order to make the structures that these modelling tools produce accessible to computational reasoning, we encode representations in a higher-order nested semantic network. This system, for which we have developed a preliminary prototype in LISP, can represent both the content of what people say, and the dynamic reasoning steps that move from one step to the next.

CCS Concepts • **Computing methodologies** → *Philosophical/theoretical foundations of artificial intelligence*;

Keywords Arxana, knowledge representation and reasoning, inference anchoring theory, conceptual dependency, mathematics, natural language, formal proof, exposition

ACM Reference format:
Joseph Corneli, Ursula Martin, Dave Murray-Rust, Alison Pease, Raymond Puzio, and Gabriela Rino Nesin. 2017. Modelling the Way Mathematics Is Actually Done. In *Proceedings of FARM'17, Oxford, United Kingdom, September 9, 2017,* 12 pages.
https://doi.org/10.1145/3122938.3122942

1 Introduction

Building a computer system that can understand mathematics at a high level was one of the intriguing challenges identified early on in the field of artificial intelligence [62, 69]. Simply understanding mathematics at all is a challenge faced by schoolchildren and others everywhere. Mathematics as it is practiced professionally is a compelling combination of formal reasoning and informal exposition. Teaching the craft of mathematics involves helping students learn how to think like experts [11], however our understanding of expert thinking in mathematics is not yet systematic. Mathematical AI could make both education and research more effective.

And yet, most mathematicians are completely uninterested in "computer mathematics" per se. The vast majority do not write their theorems in any of the many systems available for specifying and checking proofs formally. Nevertheless, the power of computational approaches has been amply demonstrated in formal mathematics [26] and in quasi-formal domains ranging from Chess and Go to logistics.

Mathematical AI could ultimately be as indispensable in conceptual domains as machine learning is in subsymbolic domains. And indeed, machine learning is likely to be useful for building mathematical AI—but in order to get to the point where such an application of machine learning techniques is possible, we would need representations of mathematics in which meaningful patterns can be found. This is the challenge towards which we address this paper.

At the moment, natural language representations of mathematics are largely obscure to text processing techniques, although inroads have been made in understanding the linguistic features of some aspects of mathematics [15, 16, 21, 22]. Mathematical formalisms take this to an extreme, since they simplify and tightly constrain the ways in which their users can write things down. We note that machine learning methods have been applied to corpora of formalised mathematics to good effect [30, 31, 33, 44]. However, mathematics as it is done by most mathematicians has a very different structure – and there is a lot more of it available, as summarised in Table 1.[1]

FARM'17, September 9, 2017, Oxford, United Kingdom

ACM ISBN 978-1-4503-5180-5/17/09…$15.00
https://doi.org/10.1145/3122938.3122942

[1] Compare: 1253 articles in the Mizar Mathematical Library, http://mmlquery.mizar.org/. Mizar is one of the oldest proof checking systems. The Archive of Formal Proofs associated with another prover, Isabelle, comprises some 362 entries (https://www.isa-afp.org/statistics.shtml).
[2] https://mathoverflow.net/questions
[3] https://math.stackexchange.com/questions
[4] https://arxiv.org/archive/math
[5] https://tools.wmflabs.org/enwp10/cgi-bin/list2.fcgi?run=yes&projecta=Mathematics

791,310	questions about mathematics in relevant Q&A forums on Stack Exchange[2,3]
292,516	mathematics papers on the Cornell e-print arXiv[4]
15,983	Wikipedia articles on mathematics[5]
25,436	books about mathematics in the Library of Congress[6]

Table 1. Sources of mainstream mathematical writing

The Stack Exchange corpus alone is much larger than the 160000 games in AlphaGo's initial training set [34], but the data looks completely different. Go games are comprised of sequences of black and white stones placed on an otherwise uniform 19×19 grid. The space of possible configurations for mathematics is much more complicated. Machine learning has also made impressive progress in textual domains – and incidentally, this remark includes DeepMind [27], the company behind AlphaGo – but the biggest successes, such as machine translation, throw away fine structural details that would be essential for machine understanding of mathematics. Coarse models of non-mathematical dialogue are state of the art for machine learning [71].

To begin to model mainstream mathematics in a way that exposes its structure to machines, we need the tools that are in some ways closer to the scholarly apparatus of literary criticism than to the formal logic used by Russell and Whitehead to write *Principia Mathematica.* Indeed, we must stress that our focus is on the way mathematical reasoning is communicated, rather than on the physical, embodied, and cultural intuitions that underlie mathematical thought. These intuitions, we believe, would be difficult to model computationally [63]. However, when they are communicated they are referred to a distinct set of reasoning patterns – albeit not only deductive, but also abductive, inductive and heuristic. Peirce said: "all mathematical reasoning is diagrammatic" [54, p. 47] – and our approach pursues this line of thinking.[7] In the AI subfield Knowledge Representation and Reasoning (KRR), the keywords "ontology" and "semantic network" denote specific types of diagrams that computers can read. We will describe a more general strategy for representing and reasoning about mathematical texts. Our approach necessarily begins *in media res* (cf. [56, p. 4]). More specifically, we put inspiring developments in formal proof, embodiment, linguistics, and machine learning each to one side for now: not because these topics are unimportant, but precisely because they are well-studied elsewhere.

In overview: we select a representation model called Inference Anchoring Theory+Content [12, 58] as the foundation of our strategy. We are not aware of any other representation framework that is both flexible and expressive enough to connect the formal and informal reasoning that is present in mainstream mathematical texts and dialogues. Our survey in §3 will point to structured proofs [40] and Lakatos Games [53] as two pieces of exemplary work that do not possess this bridging property. We also draw on Conceptual Dependence theory [45, 61, 64], and broadly, on a reflexive strategy for representation, because we are not only aiming to represent reasoning, but also to simulate it computationally – whereas IATC by itself is purely about representation. Herein we present a preliminary proof-of-concept operationalization of *flexiformalism* [36], the key tenets of which are:

(1) Formal models should show the correspondence with informally-given problems;
(2) Proofs are fundamentally informative communication;
(3) Formality is not all-or-nothing.

In outline: §2 characterizes mathematical texts in terms of their distinct formal and expository registers, and offers a high-level characterization of the current effort. §3 surveys related work, with a focus on the available modelling languages that bear on our representational goals. §4 illustrates, informally, how mathematical text can be modelled in one of these languages, and gives an initial description of our strategy for representing non-deductive reasoning. §5 adds more detail to the modelling approach and shows how existing AI strategies can be applied to make non-deductive reasoning explicit. §6 describes a prototype software implementation that we have developed in LISP, and shows how it can be applied to make the kind of diagrammatic reasoning we describe in the paper actionable on a computer. The system implementation itself is available as literate program [13]. The application to mathematics should be understood to be at a preliminary stage of development, the discussion in the current paper focuses on describing how it can be done; our argument is supported by initial experimentation. §7 presents our conclusions and returns to the four themes we set aside earlier to give an outlook on future work.

2 Background

Since our purpose is to understand mathematical reasoning as practiced by mathematicians, we began by studying examples of mathematical texts. In this section, we will summarize some high-level observations.

One of the first features which leaps to the eye when reading mathematics is that much of the text is written in a fashion which employs a precise technical vocabulary, logical

[6] https://www.loc.gov/books/?fa=partof%3Acatalog|subject%3Amathematics&all=true

[7] Peirce advanced this definition: "By diagrammatic reasoning, I mean reasoning which constructs a diagram according to a precept expressed in general terms, performs experiments upon this diagram, notes their results, assures itself that similar experiments performed upon any diagram constructed according to the same precept would have the same results, and expresses this in general terms" [54, pp. 47–48].

idioms, and symbolic notation to state mathematical propositions. We will refer to this as the *formal register* of mathematical discourse.[8] One example of an utterance in the formal register is "Every integer equals the sum of four squares." Symbolic paraphrases can readily be found, in whatever formal system one chooses. For instance, the four-square theorem could also be written $(\forall n \in \mathbb{N})(\exists m_1, m_2, m_3, m_4 \in \mathbb{N})\, n = \sum_{i=1}^{4} m_i^2$. Furthermore, in mathematical writing it is understood that nothing essential is lost in translating between the verbal and symbolic statements.

While statements in the formal register are an important part of mathematical writing, a text consisting only of formal statements would be frustrating to read because, while it may contain all the technical information, it would offer no guidance to the reader as to where the statements came from, why they are interesting, and how to go about understanding them. To offer this guidance, a mathematical text will also contain expository statements such as the following:

> Next, we will prove the four-square theorem using an algebraic identity similar to the one we just used to prove the two squares theorem.

This *expository register* has several salient features which work together and which distinguish it from the formal register.[9] While the subject matter of expository statements is also mathematical objects and propositions, the language involves not only description, but also narration and argumentation. In the example given above, we see a narrative structure in which a proof is introduced by comparing the technique to that used in a previous proof.

Whereas in formal logic, only the strictest deductive reasoning is allowed, mathematical exposition also makes use of inductive and abductive reasoning, and even looser reasoning by analogy. For instance, our example above makes an analogy between two "similar" algebraic identities. In the following sequence of questions, we see evidence of an inductive mode of reasoning in which the questioner examines several examples, searching for a suitable generalization of the problem under consideration.

> Can we do this for $x + y$? For e? Rationals with small denominator?

A related point is that, whereas formal statements only make use of the truth values and predicates of formal logic, the expository register also makes use of loose, heuristic, judgements of plausibility. Oftentimes, these are expressed using adjectives like "superficial" and "deep" which are not formally defined but refer to approximate notions which inform heuristic choices.

The expository register is also metamathematical, discussing not only objects and propositions, but also proofs and strategies. When setting out to prove a theorem, one may first assess different possible proof strategies using heuristic judgements like "depth" or "difficulty," or by making analogies with known techniques to prove abstractly similar results. In this way, informal reasoning frequently serves to guide the construction of a formal proof. Quite often, even if known, the exact deduction of a mathematical result might be lengthy and/or opaque. In this case, it is helpful to the reader to augment formal reasoning with informal reasoning which is shorter and easier to understand.[10]

In this essay, our main focus will be to construct a theory of mathematical exposition which will account for the features noted above, and implement it computationally.

2.1 Framing the current effort

For a high-level view of how mainstream mathematics relates to other challenge problems for computational understanding, one can compare previous research on the blocks world [70], board games, and story comprehension (Table 2). These require increasingly sophisticated patterns of *inference*, *thinking*, and *reasoning*, which we understand, after David Moshman [49], to be defined as follows:

- **Inference**—going beyond the data—is elementary and ubiquitous.
- **Thinking** is the deliberate application and coordination of one's inferences to serve one's purposes.
- **Reasoning** is epistemically self-constrained thinking [...] with the intent of conforming to what [are deemed] to be appropriate inferential norms.

We assert that, understood as a computational challenge, mainstream mathematics lies somewhere in between board games and story comprehension. It deals with rules (axioms) and strategies (as described, e.g., by Pólya [55]), but it also

[8] Carnap defined the *formal mode of speech* similarly: "A theory, a rule, a definition, or the like is to be called formal when no reference is made in it either to the meaning of the symbols (for examples, the words) or to the sense of the expressions (e.g. the sentences), but simply and solely to the kinds and order of the symbols from which the expressions are constructed" [10, p. 1].

[9] Carnap opposes his "formal mode" (see Note 8, above) to what he calls the *material mode of speech*. This conception is related to but distinct from what we have termed the expository register. Carnap's material mode includes "those logical sentences which assert something about the *meaning*, *content*, or *sense* of sentences or linguistic expressions of any domain" [10, p. 285]. He points out that the material mode of speech is "transposed" insofar as "in order to assert something about an object a, something corresponding is asserted about an object b which stands in a certain relation to the object a" (p. 308). In particular, "in order to say something about a word (or a sentence) we say instead something parallel about the object designated by the word (or the fact described by the sentence, respectively)" (p. 309).

[10] Lamport [40, p. 20] quotes a referee who had read one of his structured proofs (see Section 3.1.3, below): "The proofs ... are lengthy, and are presented in a style which I find very tedious. I think the readers ... are going to be more interested in understanding the techniques and how they can apply them, than they will be in reading the formal proofs. A problem with the proofs is that they do not clearly distinguish the trivial manipulations from the nontrivial or surprising parts. ... My feeling is that informal proof sketches ... to explain the crucial ideas in each result would be more appropriate."

Table 2. Comparison between computation in different domains

Level	**Blocks World**	**Board Games**	**Story Comprehension**
elements	blocks on a table	game pieces on board	episodes from everyday life
inference	follow instructions	rules & strategy	analogy
thinking	consistency	prediction of winning	costs and benefits
reasoning	(trivial)	multiple strategies	ethical dilemmas

must make sense of informal descriptions, loose parallels, and multiple points of view.

Since a city is not a tree [2], tree-shaped guides – like information scientists' ontologies, or other taxonomic descriptions – will miss its interacting sub-systems. Similarly, we claim that a formal proof will miss out the ontogenetic aspects of proving (cf. [5, p. 219]).

In the following section we draw on several strands of related work to formulate a novel approach to representation of and reasoning about mathematics.

3 Survey of related work

In the theoretical side of our work we will draw on two primary frameworks: *conceptual dependence theory*, which originated in early work in story understanding by Schank and his coauthors [61], and *inference anchoring theory*, which is a relatively recent development in the field of argumentation [6]. In what follows we contextualise our use of these two theoretical models, but do not attempt a full survey of story understanding and argumentation theory, both of which are subjects of considerable ongoing attention.[11]

The practical aspects of our work are rooted in a graphical style of reasoning. In broad terms we reason about triples, rather like those found in the Semantic Web's Resource Description Framework (RDF). Triples stand for labelled relations between objects, as in a classical semantic network. Because we need to reason about both the contents and structure of graphs in detail, relations are themselves treated as first-class objects: that is, in Semantic Web terminology, the triples are "reified" [47, §4.3]. We have experimented with different strategies for representing this data: for instance, one earlier prototype used a MySQL database that replicated simple triple store features, and we have experimented with more complex storage mechanisms. In the work presented here, we opted to use a representation strategy more closely coupled with LISP. We are broadly inspired by the genre of *annotative programming*, in which programs take the form of hypertext and dependencies are indicated by links, as pioneered in the Flare Programming Language[12] and Nelson's ZigZag [50]. In a case of parallel invention, another platform called AtomSpace has been created based on similar design decisions [23, 24]. Both AtomSpace and our system, Arxana (described in Section 6) treat relations as first-order objects, support the creation of programming structures in their graphs, and generally prefer expressibility to computational efficiency. AtomSpace directly incorporates a range of advanced features, such as attentional weighting and relations between an arbitrary number of nodes, which are not present in Arxana, but which could be simulated within our simpler "first order" relational system. AtomSpace seems a considerably more mature piece of software, e.g., it comes with both Scheme and Python bindings and a range of optimisations, and it has been used in interesting applications [43]: however, its greater complexity makes it less immediately useful as an illustrative tool for prototyping. Some of its features may represent future development targets for our system.

Having presented an overview of the landscape within which our work sits, the following subsections continue the survey, focusing in on several key landmarks that will guide our work later in the paper.

3.1 Models of mathematical reasoning

Bundy has argued that finding the right representation is the key to successful reasoning [8, p. 16]. Since we cannot simply build on formal theorem proving for our current purposes, we need to look further afield for foundations. We are concerned in particular with building representations *of* reasoning [12]. This helps guide our search for useful paradigms. McCarthy [48] posited several high-level features of a hypothetical "language of thought", highlighting in particular the ways in which such a language would not be like spoken language. We quote two key points:

- Much mental information is represented in parallel and is processed in parallel.
- Pointers to processes while they are operating may be important elements of its sentences.

Diagrammatic languages are in general well aligned with the first point. Most modern programming languages have a notion of concurrency, addressing the second. There have been several efforts to build "graphical" programming languages (e.g., [1]), but they are not directly relevant to our application. Here, we will review the features of several representation languages that we can build upon.

[11] Briefly, computational research on stories continues in many venues; one recent paper with some similarity to Schank's work is [57]. The intersections of argumentation with computer science have been explored, for example, in the biennial Computational Models of Argument conference since 2006.
[12] http://flarelang.sourceforge.net/

3.1.1 Inference Anchoring Theory + Content

Our modelling approach builds on *argumentation theory*; in particular Inference Anchoring Theory (IAT), which was devised by Budzynska et al [6, 7] to model the pro-and-contra logical structure of dialogical arguments. IAT has since been adapted to model mathematical arguments with an extension we call "IAT+Content" (IATC) [12, 58]. The primary features of this adaptation are: (1) to introduce explicit relationships between pieces of mathematical content; and (2) to describe a range of intermediate relations that model the way these objects fit together in discourse. These intermediate relations include inferential structure, judgements of validity or usefulness, and reasoning tactics. Whereas, as its name suggests, IAT anchors logical inferences in dialog norms, IATC simultaneously connects logical inferences to contentful purposes and constraints (see Moshman's definitions of *thinking* and *reasoning*, quoted above).

3.1.2 Conceptual Dependence

Conceptual Dependence was initially introduced as a tool for understanding natural language [60]. However, it is also used to represent knowledge about actions [45, 64]. An example shown in Figure 1 shows how "John sold Mary a book" can be represented as a combination of smaller actions.

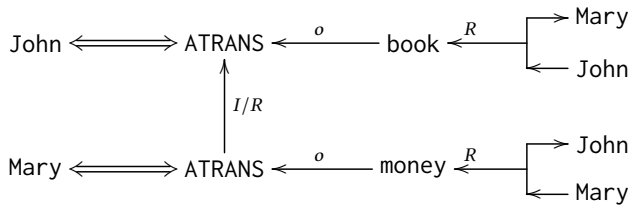

Figure 1. CD example (adapted from Figure 3 in [45])

CD has been used to reason about a range of informally-constructed scenarios. For instance, the system might be presented with a description of a restaurant in the form of a script, and a CD representation of a story such as:

> John went to a restaurant. He sat down. He got mad. He left.

Schank and Abelson's Script Applier Mechanism (SAM) could produce a paraphrase and make inferences such as concluding that John was upset because the waiter did not come [4, pp. 306 et seq.]. Furthermore, by augmenting the knowledge base with goals and plans, it became possible to explain actions. For instance:

> Willa was hungry. She picked up the Michelin guide.

Here one could explain why she picked up the guide as follows: The first sentence suggests the goal of finding food, and the restaurant script describes a way of satisfying this goal – but applying the script requires the user to know the location of a restaurant, hence the guide.

3.1.3 Structured proofs

Lamport's notion of a "structured proof" [38] is an interesting example of a "missing link" between informal and formal mathematics. In essence, a structured proof combines a high-level summary (or "proof sketch") with a mode of writing out details in such a way that each step is made clear and each assumption and assertion is clearly labelled. Structured proofs are trees, and, as such, are directed to the formal register. As first introduced, they made use of just a few keywords: `assume`, `prove`, `let`, and `case`. In a more recent reassessment [40], Lamport states "The way I now write proofs has profited by my recent experience designing and using a formal language for machine-checked proofs." In line with this, additional keywords `new`, `suffices`, and `pick` are imported from his "Temporal Logic of Actions+" (TLA$^+$) language [39], along with a shorthand for equational proofs.[13] At least these basic keywords, or equivalents, should be supported in the lower, content-addressing, layers of our language.

3.1.4 Lakatos Games

Arguments are similar to games, insofar as both typically have winners and losers: this is made precise with the notion of a *dialogue game* [9, 46].

Lakatos [37] developed an abstract, dialectical, model of mathematical creativity which has recently been formalized as one such dialogue game [52, 53]. In the "Lakatos Game", assertions are made and then progressively refined through discussion and debate. For instance, a statement may be sharpened so that it applies to a restricted class of objects ("Strategic Withdrawal"), or a putative counterexample may be shown not to be a counterexample at all ("Monster Barring"). Lakatos's theory is limited, first, in its theoretical commitments, insofar as Lakatos's mathematics is closely linked to the Hegelian notion of thesis/antithesis/synthesis [42]. Secondly, even though Lakatos's theory is now implemented computationally (per [53]), the model leaves out concrete details of mathematical objects and relationships among them. The trace of a Lakatos Game does include content – but the relationships between elements are mediated by a controlled vocabulary of 20 dialog moves, whose sequence is structured into a game tree.[14] We would like to broaden our understanding to include, for example, dialogs that do not follow strict

[13] More recently, a second version of this language called TLA^{+2} has been made available [41]. In full, TLA^{+2}'s keywords are: `action`, `assumption`, `axiom`, `by`, `corollary`, `def`, `define`, `defs`, `have`, `hide`, `lambda`, `lemma`, `new`, `obvious`, `omitted`, `only`, `pick`, `proof`, `proposition`, `prove`, `qed`, `recursive`, `state`, `suffices`, `take`, `temporal`, `use`, `witness`.

[14] Part of an example from [53, §A.1]: *Conjecture*('statement holds in the convex polygon plus point case') →*GlobalCounter*('equilateral triangle plus point', 'statement holds in the convex polygon case') →*MonsterBar*('equilateral triangle plus point', 'statement holds in the convex polygon case', 'line is alternating') →*PDefinition*('equilateral triangle plus point', 'line is alternating', '"line" extends in both directions') →*MonsterAccept*('equilateral triangle plus point', 'line is alternating').

dialectical outlines, as well as broader informal patterns of reasoning.

3.2 The search for the 'quantum of progress'

The Polymath project aimed to answer the question "is massively collaborative mathematics possible?" This project was convened by Timothy Gowers in 2009, and he developed 15 rules to address the "questions of procedure" that he anticipated would arise in the project.[15] The basic premise was that people would work together on a shared blog to discuss a given mathematical problem in the open, and jointly work out a solution. Within these ground rules, Gowers carefully explicated the imperative to share comments that are "not fully thought out." good contribution to the project would comprise what he called a "quantum of progress." Continuing with Gowers's physics metaphor, it should be clear that we need to better understand the "fields" that generate these quanta! That is, in order to build a functional representation of mathematical reasoning, we need to model both what's there at any given step, and also how it evolves – noting that such evolution is generally heuristic. To date, twelve Polymath projects and four MiniPolymath projects (focused on collective solutions to pre-college level competition problems) have been convened.[16] For researchers studying mathematical practice, these provide a useful source of data.[17]

To add to the body of thought on "proof and progress in mathematics" [68], it is interesting to compare Polymath with Gowers's joint work with Ganesalingam on a "human-oriented theorem proving" system that they call ROBOTONE [20]. Work on this system motivated Gowers's public presentation of a mathematical challenge problem that we will take up as our central example in Section 4. However, Ganesalingam and Gowers's system was only able to solve more straightforward "textbook" problems, in which the next step is always more or less "obvious." ROBOTONE has a number of heuristics, ranked by attractiveness, that it uses to transform its goal step-by-step in a logically-sound manner. ROBOTONE's progress is distinctly mechanical. The novel and interesting aspect of the system is that it generates proofs in natural language and also makes its internal processing explicit – so it straightforward to diagram its operations in IATC.[18] (However, we should emphasize that ROBOTONE's inferences are not "driven" by natural language as its main motivation or mediator.)

Sussman gives an example of a LISP program that models "'how to' as well as 'what is'" [66] – namely, a program that he wrote with with Richard Stallman that can replicate the process of analyzing a circuit [67]. The program explains *why* it comes up with the answers it does. The "why question" is even more important (and interesting) when nontrivial "creativity" is involved in the reasoning process [25]. Mathematical creativity may prove to be both possible and interesting to model, and it may be possible to devise computational systems that offer thought-provoking answers to the "why question." However, in the case of Ganesalingam and Gowers's ROBOTONE, described above, the answer to the "why question" will always be more or less the same [20, pp. 261–261]:

> the program can [...] be regarded as repeatedly applying a single tactic, which is itself constructed by taking a list of subsidiary tactics and applying the first that can be applied.

This is unlikely to work outside of a limited domain of "textbook" problems – for example, it would stumble over Polymath-style dialogues, or even "challenge problems" from outside the textbook genre – such as the example we describe in the following section.

4 Example: a diagrammatic model of mathematical reasoning via IATC

An informal proof is in general not a tree, even if a structured proof – or the trace of moves made in a dialogue game – does rather satisfactorily *describe* the proof as a tree. In the limit of full formality, tree-shaped descriptions are machine checkable – and by the Curry-Howard correspondence, proofs-as-programs are even runnable. However, by default these objects give very little information about the proof's (or program's) genesis. Indeed, as we shall see, we soon run into problems even with a still-more-general network model. Figure 2 presents an IATC analysis of Gowers's walkthrough solution to the problem "What is the 500th digit of $(\sqrt{2}+\sqrt{3})^{2012}$?" In IATC, utterances expand to *performatives* (here, only `Assert` and `Suggest` are used).[19] The performatives, in turn, expand to nodes that convey inferential structure, reasoning tactics, and heuristic judgements (e.g., `implies`, `strategy`, and `generalize`, respectively) – as well as content-level relations (e.g., "`contains as summand`"). Intermediate nodes typically have targets in the content layer. For example, the assertion that "'$(\sqrt{3}-\sqrt{2})^{2012}$' `has property` 'is small'" is typical in this regard.

The `implements` node in Figure 2, and the `implies` node somewhat lower down, both with bold outbound arrows pose some difficulty – insofar as these outbound arrows do not have another node as a target. What the `implements` node is saying is that the subgraph it is pointing at implements a heuristic that was proposed earlier, namely "The trick might be: it is close to something we can compute." Without such

[15] http://gowers.wordpress.com/questions-of-procedure/

[16] http://michaelnielsen.org/polymath1/index.php?title=Main_Page

[17] E.g., Footnote 14 draws the data that the Lakatosian framework models from one of the MiniPolymath dialogues, and Rino Nesin tagged two MiniPolymath dialogues into IATC in detail as an initial check on the completeness of the language as it was being developed.

[18] http://metameso.org/ar/robotone-example.pdf.

[19] Austin [3] characterises performative utterances as follows: (a) they do not 'describe' and accordingly have no truth value; and (b) the utterance itself performs an action.

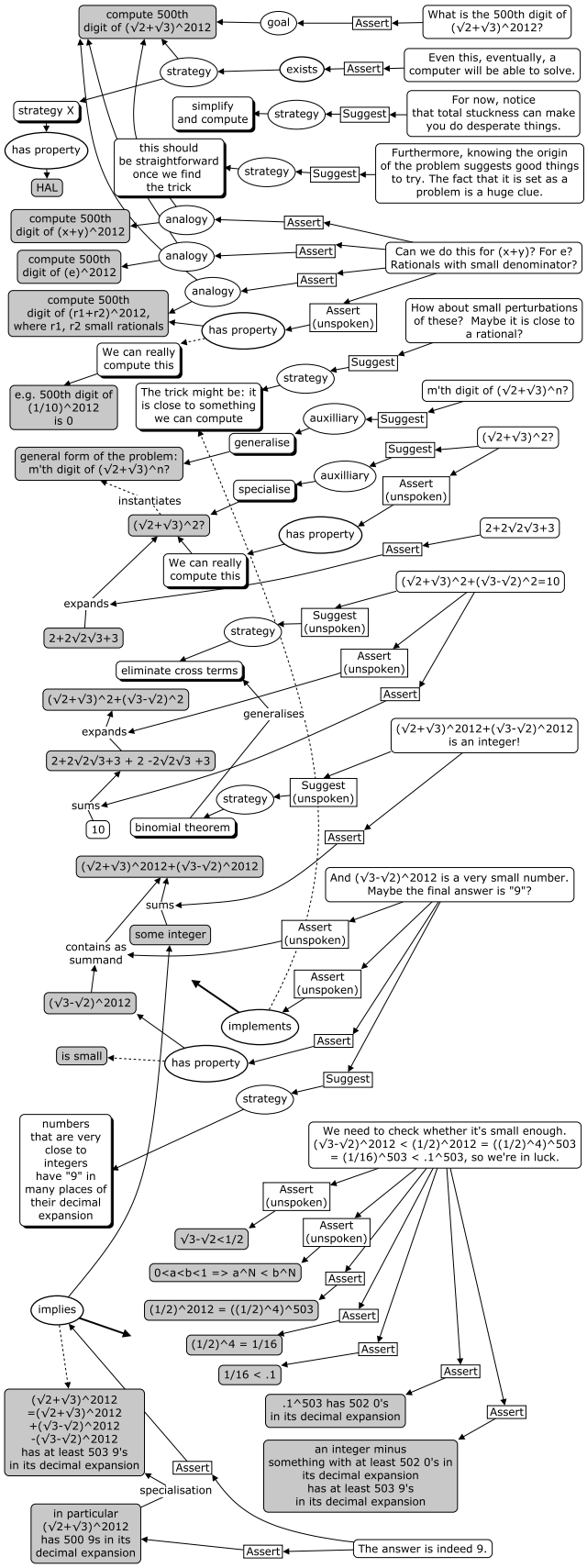

Figure 2. A mathematical challenge problem

certified implementations, we would see a proliferation of heuristic suggestions, without any sense of which ones are actually used later on. Again, this assertion is made about a subgraph, and not a specific piece of content. Recall McCarthy's assertion about the language of thought: "Pointers to processes while they are operating may be important elements of its sentences." Whereas a structured proof would just give the indicated subgraph a name and push it down the hierarchy into a lemma, we can see from the diagram that such a representation strategy has limitations. The inbound link to "some integer" (from the subsequent `implies` reasoning step) could be represented formally as a reference to the lemma's conclusion, but the link to an heuristic introduced much earlier on is entirely informal. The example here is simple, but reasoning about graph structure has the potential to become highly complex. To cope with these complications, we need *hypergraphs*, not just graphs. Specifically, we will rely on one of the standard generalizations of the hypergraph concept: the ability for an edge to point to another edge.[20] The basic reason, again, is that we need a way to lasso the embedded "lemma" as an object of interest, and point to it, without wholly separating it off from the rest of the proof (see Section 6).

5 Towards functional models of mathematical reasoning

By combining ideas from IATC (for representation) and CD (for reasoning) it should be possible to have the system look at the data in Figure 2 and answer questions like:

- "Why are we interested in integers at this point?"
- "Why is 9 seen as a likely answer once we know that $(\sqrt{3}-\sqrt{2})^{2012}$ is small?"

The premise here is that the dynamics at work are based on performatives being chosen based upon current valuations, goals, and other contextual features. We expand this idea in the following subsections.

5.1 Kinematics (IATC+CD)

The basis of CD theory is a classification of actions into broad classes called *primitives*, each of which are equipped with a number of places for arguments (referred to as *slots*) which hold objects that fulfil roles specific to that class of actions. When a primitive's slots have been filled in, the result takes the form of the graph.

We emphasize that the primitives of CD were designed to represent everyday activities in the physical world and hence might not be the best fit for mathematical texts. Therefore, we will instead use the IATC performatives as our action primitives. Like the standard CD primitives, these performatives have slots. The statements that are introduced via these performatives also have slots, e.g., '`has_property` (*o*,

[20] https://www.slideshare.net/delaray/knowledge-extraction-presentation: "Mom resents the fact that John disapproves of Jane and Jim's marriage."

p)' asserts that object *o* has property *p*. The specification of IATC includes 9 performatives and 15 intermediate relations [58]; a subset of the specification is given in Appendix A.

Assertions that are fully unfolded generally bottom out in the content layer: assertions can be made directly about objects and propositions, or about relationships between the same, as in Figure 2's "`contains as summand`" relation; content can be addressed in other intermediate assertions, such as "$(\sqrt{3}-\sqrt{2})^{2012}$" `has_property` "is small". The performatives rooted on the utterance

> And $(\sqrt{3}-\sqrt{2})^{2012}$ is a very small number. Maybe the final answer is "9"?

are represented as s-expressions in Listing 1. (We defer further comment on the somewhat technical matter of pointing at a specific subgraph to Section 6, but highlight here that that from the point of view of IATC, it just means that we can substitute "subgraph" for "statement" in slots that ask for a statement.)

```
(Assert
 "contains as summand"
 "(sqrt(2)+sqrt(3))^2012+(sqrt(3)-sqrt(2))^2012"
 "(sqrt(3)-sqrt(2))^2012")

(Assert (has_property "(sqrt(3)-sqrt(2))^2012"
                      "is small"))

(Assert (implements #SUBGRAPH
                    "the trick might be: it
                     is close to something
                     we can compute"))

(Suggest (strategy "numbers that are very close
                    to integers have \"9\"
                    in many places of their
                    decimal expansion"))
```

Listing 1. Assertions from Figure 2 modelled with CD-like code, using IATC performatives as our primitives

5.2 Dynamics

Static graphical representations of mathematical reasoning do not by themselves offer *functional* models of mathematical reasoning. We need to be able to reason about structure: answering "why" questions, as above, or introducing new reasoned statements into a dialogue. Here we are inspired by Sowa and Majumdar's treatment of reasoning by analogy [65]. For example, early in the solution to the challenge problem from Figure 2, Gowers asks:

> Can we do this for $x + y$? For e? Rationals with small denominator?

Inspecting the context, we see that the anaphor "this" refers to *computing the 500th digit of X*. As we remarked in Section 2, the quoted sequence of questions embodies an inductive search for a suitable generalization of the question; specifically, this search arrives at "the mth digit of $(\sqrt{2}+\sqrt{3})^n$". Sowa and Majumdar [65, §2] show that inductive reasoning and analogical reasoning can achieve similar results. Using a typed formalism we could be specific about the difference between the two sides, while retaining a structural similarity on the two sides of the analogy.

Given sufficiently detailed representations – sometimes requiring background theories – we expect analogical reasoning to serve as the main explanatory mechanism. Sowa and Majumdar describe three ways to do analogical reasoning about conceptual graphs. There is in fact a (high-order) analogy between what goes on in building representations via parsing and reasoning about the representations so-described. Lytinen [45] summarises Schank and Birnbaum [59] on semantic parsing, again, referring to three aspects of the parser: Making this analogy more precise, Sowa and Majumdar refer to a "graph grammar"; this topic is developed in more detail by Ehrig et al [18]. We will describe an example relevant to our setting in the following section.

Originally, the various systems related to CD were implemented in different software packages such as SAM, PAM, and VAE [4, 65]. Rather then adapt these packages and interface them, we will take a different approach: computing with hypergraphs.

6 Our prototype, *Arxana*

6.1 System overview

In order to store graphical representations of knowledge in memory and query them, we will make use of a system called *Arxana* which we have been developing in LISP [13]. The basis of this system is higher-order nested semantic networks. By "higher order," it is meant that links can point to other links. By "nested," it is meant that the nodes which make up a network may contain other networks, which themselves may have nodes containing yet other networks, etc. In addition, all the links are bidirectional and no fundamental distinction is made between links and nodes. We call the basic unit from which we will construct our networks a *nema*. Nemas (or nemata) are data objects which serve to encode both links and nodes and are characterized by the following components: **Identifier**, **Source**, **Sink**, and **Content**. Source and sink are pointers to other articles and content is a place in which to store a LISP object, which might be some text, or an expression, or maybe a number or maybe something else. In particular, the content of an article can even be another network constructed out of more articles. By choosing suitable conventions, one can build up more complex functionality from the basic features. For instance, one can introduce typing by designating a certain nema as the "home nema" for that type and interpreting a link whose source is the home

nema and whose sink is some other nema as stating that the sink nema is of that type.

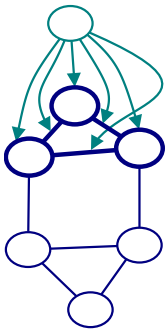

Figure 3. Cone over a subgraph

Likewise, as shown in Figure 3, one can encode subhypergraphs by designating a specific nema as a label, and then indicating the components of the hypergraph of interest by links from that label. Here, the lighter-colored nema at the top of the diagram together with the arrows emanating from it point out a bold-faced triangle comprised of three nodes and the lines between them. We call this structure a *cone*.[21] We could use a similar construction to make the `implements` relation discussed in Section 4 explicit, namely by pointing to all of the constituent elements of the relevant subgraph.

In order to make these representations above actionable, we use two main facilities: *hypergraph matching* and *hypergraph programming*. Given a hypergraph, we can query it for subhypergraphs which match some suitable criterion. A query consists of a hypergraph whose articles contain predicates and a match consists of a mapping of hypergraphs from the query onto a subhypergraph such that, for every article in the query, the predicate it contains is true of the contents of the corresponding database article.

Like a cons cell, a nema comes with two links, *source* and *sink* (analogous to car and cdr) which point to other nemata. However, we extend the model in two ways: firstly, each nema also contains an additional place, called *content*, in which one can store an arbitrary LISP object which could be a text string, a program, or even a network. By embedding programs inside the hypergraph, reasoning steps can be stored close to where they are used. In particular, we can store hypergraph grammar rules that are used to transform graphs (e.g., on a link).

6.2 Application to mathematical reasoning

Relative to our current purposes, the first key function of Arxana is to make graphical representations like the ones in Figure 2 accessible to the computer.

Here, we focus in on expanding and making sense of the s-expressions from Listing 1. Note that the utterances recorded in Figure 2 do not contain everything needed for this task. In particular, some of the expansions that shown in the figure are noted to be "unspoken". Background information about mathematics needs to be programmed into a knowledge base, i.e., the system needs a range of facts, so that, for example, "is small" is known to be synonymous to "close to zero". Relatedly, the lengthy strings embedded in Listing 1 need to be parsed further – for example "the trick might be: it is close to something we can compute" breaks down to two content-level statements: "it is close to something" and "something is computable", along with a meta-level bid to find suitable objects with the corresponding properties.

When we expand the "`implements`" and "`#SUBGRAPH`" we obtain a node whose content is a proof certificate and a cone below that node, respectively, as illustrated in Figure 2 and described above. What's missing from Figure 2, however, is a link between this certificate and the new strategy the same set of performatives `Suggests`. A detailed IATC+CD analysis reveals that this missing link, F, emerges from the reasoning process itself. We demonstrate one part of this derivation below, leaving out a few steps related to the first s-exp from Listing 1, which a computer algebra system could quickly make sense of. The result of this process is a second certificate that is provided by the last lambda expression in Listing 2 and verifies the reasoning behind the suggested strategy. The corresponding cone (illustrated in Figure 4) is constructed from eight links labelled $A, B, C, D, E, F, \alpha, \beta, \gamma$. The links A, B, C, D, E, F point to semantic subnetworks whose serializations are given at the top of Listing 2 and the links α, β, γ point to lambda expressions which implement inference rules. These lambda expressions are of the form `(lambda (plex) (replace (car (search LHS plex)) RHS))` where `LHS` and `RHS` are subexpressions specific to the inference rule and are given in the middle of Listing 2.

Verification proceeds by using the Arxana facility for running embedded programs. When it runs, the certificate first gathers the various premises and conclusions from the cone. Starting with the premise C, it apples the rule "`weaken-equiv`" which it finds by following link γ of the cone. Implementing this rule is accomplished by hypergraph search and replacement. It then proceeds similarly with the remaining hypotheses, using inference rules to extend the chain of reasoning and finishes by checking whether the final result agrees with the conclusion pointed to by link F.

At the higher level, one would invoke an inference engine such as mini-kanren (from *The Reasoned Schemer* [19]) to unify the conclusion from the hypotheses. Analogous questions about restaurants were answered by Schank et al.'s Lisp programs, however their work predates Prolog: nevertheless, CD inferencing and parsing would port over to a logic language naturally.

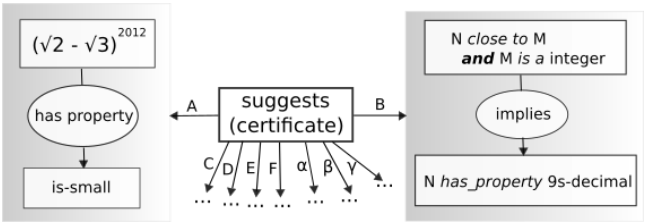

Figure 4. The reasoning itself suggests a key strategy

[21] From https://ncatlab.org/nlab/show/cone: "In homotopy theory, the cone of a space X is the space got by taking the X-shaped cylinder $X \times I$, where I may be an interval object, and squashing one end down to a point."

```
;; assertions
(A (has_property "(sqrt(3)-sqrt(2))^2012" is-small))
(B (implies (and (close-to N M)
                 (isa M integer))
            (has_property N 9s-decimal)))

;;; existing knowledge base
(C (equivalent-to (N has_property is-small)
                  (N close-to 0)))
(D (isa 0 integer))
(E (implies (N isa integer)
            (N has_property computable)))

;;; conclusion
(F (viable-strategy "(sqrt(3)+sqrt(2))^2012"
                    (has_property N 9s-decimal)))

;;; inference rules
(simplify-has_property+implies
 :lhs (((y) (a has_property) (x) (b implies) (z))
       ((x src y) (y flk a) (x snk z) (x flk b)))
 :rhs (((y) (a has_property) (z))
       ((z src y) (y flk a))))
(transitivity
 :lhs (((y) (a implies) (x) (b implies) (z))
       ((x src y) (y flk a) (x snk z) (b flk z)))
 :rhs (((y) (a implies) (z))
       ((z src y) (y flk a))))
(weaken-equiv
 :lhs (((b equivalent-to)))
 :rhs (((b implies))))

;;; computing the certificate
(lambda (V)
  (apply
   (lambda (F A B C D)
     (equal
      (simplify-has_property+implies
       (graph-union
        B
        (transitivity
         (graph-union
          D
          (simplify-has_property+implies
           (graph-union
            A
            (weaken-equiv C)))))))
      F))
   (mapcar
    (lambda (X)
      (get-sink
       (car
        (triples-given-beginning-and-middle V X)))
      '(A B C D)))))
```

Listing 2. Key elements for reasoning about Listing 1

7 Conclusions and Future Work

The formal register of mathematical discourse has been studied by de Bruijn, Ranta, Ganesalingam, and others. Once we have formal representations, it is possible to reason effectively about them using tools from Frege and others. In this paper we have focused, instead, on a computational theory of the expository register. We have drawn upon both classic and recent AI research which has considered situations with which everyday mathematics shares common features.

Quite to the contrary of the sentiment conveyed by the misattributed quote "Mathematics is a game played according to certain simple rules with meaningless marks on paper," David Hilbert, an early and important proponent of mathematical formalism, emphasized the role of intuition and experience, and discussed conjectural thinking and the fallibility of mathematical reasoning [14, 28].[22] Our work continues roughly in this spirit: but rather than referring intuition to logical and mathematical foundations, we have outlined a novel, if preliminary, operationalization of flexiformal representation and reasoning. Specific items of future work will re-integrate the themes we set aside in the introduction to this paper. They include:

Formal proof

- Demo the system walking through the steps of a proof like the challenge problem we discussed, or a MiniPolymath dialogue, which would necessitate refining both representations and reasoning.
- Integrate external SMT solvers, proof checkers, proof assistants, and computer algebra systems (e.g., 'X `sums` "some integer"' and 'X `contains as summand` Y' could be treated by different external systems).

Embodiment and cognitive science

- Build on CD theory to reason about embodied intuitions in geometric problems, integrate with Lakoff and Núñez's conceptual metaphors [51]; connect this with formal approaches in geometry.
- Relate reasoning about hypergraphs to work by Silvia de Toffoli [17], Mateja Jamnik [29], and other contemporary scholars working on reasoning with diagrams.

Linguistics and NLP

- Integrate parsers to generate IATC+CD automatically, initially via intermediate semantic markup [35].
- The IATC specification (presented in part above) is to be understood as preliminary, and the language should be expanded and improved based on corpus studies.
- Use statistical methods on the relevant corpora, expanding on the work of Kaliszyk et al [32] who ascertained the frequency of various schematic usages like "let X be a Y" in a specific corpus of proofs.

[22] https://www.cs.nyu.edu/pipermail/fom/2005-April/008889.html.

Machine learning

- Integrate with knowledge bases of mathematical terms and frequency data (as above).
- Model Stack Exchange dialogues, in parallel with the work done on Reddit discussions [71].
- Build a system with multiple agents that "converse with each other to sharpen their wits" [69].

8 Acknowledgements and supplement

The authors would like to thank Daniel Pique for useful discussions and advice, and Steve Corneli, who provided extensive comments on an early draft of the paper. We also thank the anonymous workshop reviewers and our paper shepherd Michael Sperber for their suggestions, which significantly improved the final submission.

The authors acknowledge the financial support of the EPSRC via Ursula Martin's fellowship "The Social Machine of Mathematics" (EP/K040251/1). The paper also benefitted from conversations at the Isaac Newton Institute residential programme on "Big Proof" (EP/K032208/1).

Auxilliary material, including computer programs, data sets, and full versions of diagrams quoted above are available via this URL: http://metameso.org/ar/.

References

[1] Tom Addis and Jan Addis. *Drawing programs: the theory and practice of schematic functional programming*. Springer Science & Business Media, 2009.

[2] Christopher Alexander. A city is not a tree. *Architectural Forum*, 122(1):58–62, April 1965.

[3] John Langshaw Austin. *How to Do Things With Words*. Oxford University Press, 1975.

[4] Avron Barr and Edward A Feigenbaum. *The handbook of artificial intelligence*, volume 1. Butterworth-Heinemann, 1981.

[5] Jean-Hugues Barthélémy. Fifty key terms in the works of Gilbert Simondon. In Arne De Boever, Alex Murray, Jon Roffe, and Ashley Woodward, editors, *Gilbert Simondon: Being and Technology*, pages 203–231. Edinburgh University Press, 2012.

[6] Katarzyna Budzynska, Mathilde Janier, Chris Reed, and Patrick Saint-Dizier. Towards extraction of dialogical arguments. In *Proceedings of 13th International Conference on Computational Models of Natural Argument (CMNA 2013)*, 2013.

[7] Katarzyna Budzynska, Mathilde Janier, Chris Reed, Patrick Saint-Dizier, Manfred Stede, and Olena Yaskorska. A Model for Processing Illocutionary Structures and Argumentation in Debates. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May 26-31, 2014*, pages 917–924, 2014.

[8] Alan Bundy. The interaction of representation and reasoning. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 469(2157), 2013.

[9] Lauri Carlson. *Dialogue games: An approach to discourse analysis*. Springer Science & Business Media, 2012.

[10] Rudolf Carnap. *Logical syntax of language*. Psychology Press, 1937.

[11] Allan Collins, John Seely Brown, and Susan E Newman. Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. *Thinking: The Journal of Philosophy for Children*, 8(1):2–10, 1988.

[12] Joseph Corneli, Ursula Martin, Dave Murray-Rust, and Alison Pease. Towards mathematical AI via a model of the content and process of mathematical question and answer dialogues. In Herman Geuvers, Matthew England, Osman Hasan, Florian Rabe, and Olaf Teschke, editors, *Intelligent Computer Mathematics 10th International Conference, CICM 2017, Edinburgh, UK, 2017, Proceedings*, 2017.

[13] Joseph Corneli and Raymond Puzio. Arxana 2017. Technical report, metameso.org/ar/, 2017.

[14] Leo Corry. The origins of eternal truth in modern mathematics: Hilbert to Bourbaki and beyond. *Science in Context*, 10(02):253–296, 1997.

[15] Marcos Cramer, Peter Koepke, and Bernhard Schröder. Parsing and disambiguation of symbolic mathematics in the Naproche system. In *International Conference on Intelligent Computer Mathematics*, pages 180–195. Springer, 2011.

[16] Marcos Cramer, Daniel Kühlwein, and Bernhard Schröder. Presupposition Projection and Accommodation in Mathematical Texts. In *KONVENS*, pages 29–36, 2010.

[17] Silvia De Toffoli. 'Chasing' The Diagram–The Use of Visualizations in Algebraic Reasoning. *The Review of Symbolic Logic*, 10(1):158–186, 2017.

[18] Hartmut Ehrig, Annegret Habel, and Hans-Jörg Kreowski. Introduction to graph grammars with applications to semantic networks. *Computers & Mathematics with Applications*, 23(6-9):557–572, 1992.

[19] Daniel P Friedman, William E Byrd, and Oleg Kiselyov. *The Reasoned Schemer*. MIT Press, 2005.

[20] M. Ganesalingam and W. T. Gowers. A Fully Automatic Theorem Prover with Human-Style Output. *Journal of Automated Reasoning*, pages 1–39, 2016.

[21] Mohan Ganesalingam. *The Language of Mathematics, A Linguistic and Philosophical Investigation*, volume 7805 of *LNCS*. Springer Verlag, 2013.

[22] Deyan Ginev. The structure of mathematical expressions. Master's thesis, Jacobs University, Bremen, Germany, 2011.

[23] Ben Goertzel, Cassio Pennachin, and Nil Geisweiller. *Engineering General Intelligence, Part 1: A Path to Advanced AGI via Embodied Learning and Cognitive Synergy*, volume 5 of *Atlantis Thinking Machines*. Springer, 2014.

[24] Ben Goertzel, Cassio Pennachin, and Nil Geisweiller. *Engineering General Intelligence, Part 2: The CogPrime Architecture for Integrative, Embodied AGI*, volume 6 of *Atlantis Thinking Machines*. Springer, 2014.

[25] Christian Guckelsberger, Christoph Salge, and Simon Colton. Addressing the "Why?" in Computational Creativity: A Non-Anthropocentric, Minimal Model of Intentional Creative Agency. In Ashok Goel, Anna Jordanous, Alison Pease, Mikhail Jacob, and Matthew Guzdial, editors, *Proceedings of the Eighth International Conference on Computational Creativity, ICCC 2017*, 2017.

[26] Thomas Hales et al. A formal proof of the Kepler conjecture. In *Forum of Mathematics, Pi*, volume 5. Cambridge University Press, 2017.

[27] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[28] David Hilbert. *Natur und mathematisches Erkennen: Vorlesungen, gehalten 1919-1920 in Göttingen*. Birkhäuser, [1919] 1992.

[29] Mateja Jamnik. *Mathematical reasoning with diagrams*. University of Chicago Press, 2001.

[30] Cezary Kaliszyk, François Chollet, and Christian Szegedy. HolStep: A Machine Learning Dataset for Higher-order Logic Theorem Proving. *CoRR*, abs/1703.00426, 2017.

[31] Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *Journal of Automated Reasoning*, 53(2):173–213, 2014.

[32] Cezary Kaliszyk, Josef Urban, Jiří Vyskočil, and Herman Geuvers. Developing corpus-based translation methods between informal and formal mathematics [Poster of [33]]. http://cl-informatik.uibk.ac.at/cek/docs/14/ckjujvhg-cicm14-poster.pdf.

[33] Cezary Kaliszyk, Josef Urban, Jiří Vyskočil, and Herman Geuvers. Developing corpus-based translation methods between informal and formal mathematics. In *International Conference on Intelligent Computer Mathematics*, pages 435–439. Springer, 2014.
[34] Christof Koch. How the Computer Beat the Go Master, March 2017.
[35] M. Kohlhase. sTEX: Semantic markup in TEX/LATEX. Self-Documenting LATEX package, 2017.
[36] Michael Kohlhase. The flexiformalist manifesto. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on*, pages 30–35. IEEE, 2012.
[37] Imre Lakatos. *Proofs and Refutations: The Logic of Mathematical Discovery*. Cambridge University Press, 1976.
[38] Leslie Lamport. How to write a proof. *The American Mathematical Monthly*, 102(7):600–608, 1995.
[39] Leslie Lamport. Specifying Concurrent Systems with TLA$^+$. *NATO Science Series, III: Computer and Systems Sciences*, 173(173):183–247, 1999.
[40] Leslie Lamport. How to write a 21st century proof. *Journal of fixed point theory and applications*, 11(1):43–63, 2012.
[41] Leslie Lamport. TLA^{+2}: A Preliminary Guide. 2014.
[42] Brendan Larvor. Lakatos's Mathematical Hegelianism. *The Owl of Minerva*, 31(1):23–44, 1999.
[43] Ruiting Lian, Ben Goertzel, Linas Vepstas, David Hanson, and Changle Zhou. Symbol Grounding via Chaining of Morphisms. *CoRR*, abs/1703.04368, 2017.
[44] Sarah M. Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep Network Guided Proof Search. *CoRR*, abs/1701.06972, 2017.
[45] Steven L Lytinen. Conceptual dependency and its descendants. *Computers & Mathematics with Applications*, 23(2-5):51–73, 1992.
[46] William C. Mann. Dialogue games: Conventions of human interaction. *Argumentation*, 2(4):511–532, Nov 1988.
[47] Frank Manola, Eric Miller, Brian McBride, et al. RDF primer. *W3C recommendation*, 2004.
[48] John McCarthy. The well-designed child. *Artificial Intelligence*, 172(18):2003–2014, 2008.
[49] David Moshman. From inference to reasoning: The construction of rationality. *Thinking & Reasoning*, 10(2):221–239, 2004.
[50] Theodor Holm Nelson. Zigzag (tech briefing). In *Proceedings of the 12th ACM conference on Hypertext and Hypermedia*, pages 261–262. ACM, 2001.
[51] Rafael Núñez and George Lakoff. The cognitive foundations of mathematics. *Handbook of mathematical cognition*, pages 109–124, 2005.
[52] Alison Pease, Katarzyna Budzynska, John Lawrence, and Chris Reed. Lakatos Games for Mathematical Argument. In S. Parsons, N. Oren, C. Reed, and F. Cerutti, editors, *Proceedings of the Fifth International Conference on Computational Models of Argument (COMMA 2014)*, pages 59–66, Pitlochry, 2014. IOS Press.
[53] Alison Pease, John Lawrence, Katarzyna Budzynska, Joseph Corneli, and Chris Reed. Lakatos-style Collaborative Mathematics through Dialectical, Structured and Abstract Argumentation. *Artificial Intelligence*, 246:181–219, May 2017.
[54] Charles S. Peirce. *The New Elements of Mathematics*, volume 4. Mouton, 1976.
[55] G. Pólya. *How to Solve It*. Princeton University Press, 1945.
[56] Willard Van Orman Quine, Patricia S Churchland, and Dagfinn Føllesdal. *Word and object*. MIT press, [1960] 2013.
[57] Mark O Riedl and Brent Harrison. Using stories to teach human values to artificial agents. In Paula Boddington, Miles Brundage, Joanna Bryson, Judy Goldsmith, Ben Kuipers, and Toby Walsh, editors, *AI, Ethics, and Society Workshop at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
[58] Gabriela Rino Nesin. Extending Inference Anchoring Theory for use with mathematical argumentation. Technical report, University of Edinburgh, 2016.
[59] R. Schank and L Birnbaum. Memory, meaning, and syntax. Technical Report 189, Department of Computer Science, Yale University, 1980.
[60] Roger C Schank. Conceptual dependency: A theory of natural language understanding. *Cognitive psychology*, 3(4):552–631, 1972.
[61] Roger C Schank and Robert P Abelson. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press, [1977] 2013.
[62] Herbert A Simon and Allen Newell. Heuristic problem solving: The next advance in operations research. *Operations research*, 6(1):1–10, 1958.
[63] Aaron Sloman. The well-designed young mathematician. *Artificial Intelligence*, 172(18):2015–2034, 2008.
[64] John F Sowa. Conceptual Graphs. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, chapter 5, pages 213–237. Elsevier, 2008.
[65] John F Sowa and Arun K Majumdar. Analogical reasoning. In A. Aldo, W. Lex, and B. Ganter, editors, *Conceptual Structures for Knowledge Creation and Communication: 11th International Conference on Conceptual Structures, ICCS 2003, Dresden, Germany, July 21-25, 2003, Proceedings*, number 2746 in LNAI, pages 16–36. Springer, 2003.
[66] Gerald Jay Sussman. Why programming is a good medium for expressing poorly understood and sloppily formulated ideas. In *OOPSLA '05: Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 6–6. ACM, 2005.
[67] Gerald Jay Sussman. We Really Don't Know How To Compute! In *Strange Loop*, 2011.
[68] W.P. Thurston. On proof and progress in mathematics. *Bulletin (New Series) of the American Mathematical Society*, 30(2):161–177, 1994.
[69] A. M. Turing. Intelligent Machinery, A heretical theory. *Philosophia Mathematica*, 4(3):256–260, [1957] 1996.
[70] Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical Report 235, MIT AI Lab, February 1971.
[71] Amy X Zhang, Bryan Culbertson, and Praveen Paritosh. Characterizing Online Discussion Using Coarse Discourse Sequences. In *Proceedings of the Eleventh International Conference on Web and Social Media*. AAAI Press, 2017.

A Partial specification of IATC

`Assert` (*s* [, *a*])	Assert belief that statement *s* is true, optionally because of *a*.
`Agree` (*s* [, *a*])	Agree with a previous statement *s*, optionally because of *a*.
`Challenge` (*s* [, *a*])	Assert belief that statement *s* is false, optionally because of *a*.
`Retract` (*s* [, *a*])	Retract a previous statement *s*, optionally because of *a*.
`Define` (*o*, *p*)	Define object *o* via property *p*.
`Suggest` (*s*)	Suggest a strategy *s*.
`Judge` (*s*)	Apply a heuristic value judgement *s* to some statement.
`Query` (*s*)	Ask for the truth value of statement *s*.
`QueryE` ($\{p_i(X)\}$. *i*)	Ask for the class of objects X for which all of the properties p_i hold.
`has_property` (*o*, *p*)	Object *o* has property *p*.
`strategy` (*m*, *s*)	Method *m* may be used to prove statement *s*.
`beautiful` (*s*)	Statement *s* is beautiful.