A Promising Path Towards Autoformalization and General Artificial Intelligence

Christian Szegedy(✉)

Google Research, Mountain View, CA, USA
szegedy@google.com

Abstract. An autoformalization system is an AI that learns to read natural language content and to turn it into an abstract, machine verifiable formalization, ideally by bootstrapping from unlabeled training data with minimum human interaction. This is a difficult task in general, one that would require strong automated reasoning and automated natural language processing capabilities. In this paper, it is argued that autoformalization is a promising path for systems to learn sophisticated, general purpose reasoning in all domains of mathematics and computer science. This could have far reaching implications not just for mathematical research, but also for software synthesis. Here I provide the outline for a realistic path towards those goals and give a survey of recent results that support the feasibility of this direction.

1 Introduction

Today, AI systems are able to learn solving tasks that used to be thought of taking uniquely human capabilities until recently: computer vision [46], generating artistic images [13], music [21], mastering the game of go [43], discovering novel drugs [15] and performing symbolic integration [31], to name just a few. These and many other domains seemed to require uniquely human intuition and insight, but were transformed by deep learning in the past few years. While progress has been extremely impressive in those areas, each particular solution addresses a relatively narrow use case. On the other hand, general reasoning still seems a uniquely human feat and many [20] would argue that creating AI agents with general reasoning capabilities equaling to those of humans would take decades, maybe centuries, if possible at all.

This invited paper argues that in the coming years we will see automated systems to rival humans in general reasoning and the fastest path to achieve this is by creating automated mathematical reasoning systems via autoformalization.

Here, I give an overview of the hurdles involved, a realistic path ahead and indications on the feasibility of that path.

Mathematics is the discipline of pure reasoning. *Mathematical reasoning is not about mathematics per se, it is about reasoning in general.* Whether to verify

https://doi.org/10.1007/978-3-030-53518-6_1

the correctness or resource use of a computer program or to derive the consequences of a physical model, it is all mathematical reasoning, as long as it is based on fully formalized premises and transformation rules. Some tasks may require such a large number of logical steps that humans find it impossible to check them manually, but often they are easily solved by SAT-solvers [5] – programs whose sole goal is to decide if a Boolean expression can ever evaluate to true.

For certain classes of expressions, like those that occur frequently in chip design, SAT solvers work remarkably well [10]. An extreme demonstration of their power is their use in the computer generated proof of a previously unsolved famous conjecture in mathematics [25] – the Boolean Pythagorean triples problem. The final proof was 200 terabytes long.

However, SAT-solvers cannot verify statements about infinitely many cases. For example, they can't even verify that the addition of integer numbers is commutative. There are automated theorem provers (ATPs [11]) for finding moderately difficult proofs in first order logic that can deal with such problems. Proof automation via "hammers" [6,27] is also applied for higher order logic as well in the context of interactive theorem proving. Most existing proof automation is based on hand engineered heuristics, not on machine learning and is not capable of open-ended self-improvement.

Mathematical reasoning is just reasoning about anything specified formally. Reasoning about anything formal could be a powerful general tool. If we want to create an artificially intelligent system and demonstrate its general intelligence, it should be able to reason about any area of mathematics or at least it should be able to *learn to do so* given enough time. If it succeeds in practice, then we can be convinced that it is likely that it will be able to learn to cope with any scientific discipline as far as it can be formalized precisely.

Human mathematics consists of a large variety of loosely connected domains, each of them having its own flavor of proofs, arguments and intuition. Human mathematicians spend years studying just to become experts in a few of those domains. An artificial system engineered to produce strong results in a particular area is not a "general purpose" reasoning engine. However, if a system demonstrates that it can learn to reason in any area it is exposed to, then that would be a convincing demonstration of artificial general intelligence.

Therefore it is natural to ask: Will we ever arrive at the point where an AI agent can learn to do reasoning as well as the best humans in the world in most established domains of mathematics.

2 What is (Auto-)formalization?

The task of formalization is to turn informal descriptions into some formally correct and automatically checkable format. Examples of mathematical formalization include the formal proofs of the Kepler conjecture [22], the Four-Color theorem [16] and the Feit-Thompson theorem [17]. These formalization works required a lot of human effort. For example the formalization of the Kepler conjecture took over 20 man-years of work. The aim of autoformalization would be to automate such efforts and scale them up to process large chunks of existing mathematics in a fully automated manner.

More generally, "formalization" can refer to any process that takes an informal description for input and produces machine executable code. By this definition, formalization covers both programming and mathematical formalization. This generalized notion is also justified because computer verifiable proofs are in fact programs to feed some minimalistic verification kernel. For example, most proof assistants are complete programming languages that allow for running arbitrary programs while guaranteeing the correctness of the produced proofs.

Complex mathematics is especially time consuming to formalize by humans. Therefore, it is highly unlikely that a significant portion of mathematics will be formalized manually in the coming decades. Could formalization be ever automated completely? The ideal solution could process natural language text fully automatically, with minimal intervention from the user.

We call an automated system that is capable of automatically formalizing significant portions of mathematics from a natural language input and verifying it automatically an *autoformalization system.*

3 Why is Autoformalization Essential?

Is targeting autoformalization a prerequisite for training – and evaluating – AI systems for general purpose reasoning?

As was argued in the introduction, all formalizable reasoning can be viewed as mathematical in nature. Conversely, general purpose reasoning systems should be able to learn to reason about *any* domain of mathematics and should be able to discover new mathematical domains when needed or useful for another task.

Avoiding autoformalization (interpreting natural language text and communicating in natural language) would seem to simplify the engineering of formal reasoning systems. However, evaluating a highly sophisticated, general purpose, automated mathematical reasoning system without natural language communication capabilities would raise several problems:

1. Training and evaluation of a purely formal system would require a wide range of formalized statements. Creating a large corpus of diverse and correct formalized statements is a daunting task in and of itself.
2. Any human interaction with our system would be by formal inputs and outputs. If the system is trained by automated exploration and develops its own web of definitions (about which it does not need to communicate in natural language), it will resemble alien mathematics that is very hard to decipher and interpret by humans.
3. Every time the system needs to be applied to a new application domain, it would require full-blown manual formalization of that domain. This would limit its usefulness significantly.

Training a strong mathematical reasoning system without autoformalization might be still possible if one could develop a concise, well defined notion of “interestingness” that is used as the objective for open-ended exploration. However it would be very hard to communicate with such a system as it would not be able to communicate in terms of human mathematics. Furthermore, “interestingness” and “usefulness” of mathematical statements and theories are not easy to capture formally. It is hard to decide whether some mathematical area will ever have external applications or would provide insights for other domains down the line. Usefulness is highly contextual. There is no known way to guide a search process automatically towards useful theorems and notions in an open ended manner.

Since only a tiny portion of human mathematics is formalized currently, the only way for utilizing a significant fraction of accumulated human mathematical knowledge is by processing it from natural language. Therefore, the safest option is to use the entirety of human mathematics as a basis for training and benchmarking.

It could be easier to engineer and train an AI agent that can reason and formalize at the same time than designing one for just reasoning or just for formalization alone if one manages to initiate a positive feedback loop between reasoning abilities and formalization capabilities. Improving one aspect (translation or reasoning) of the system helps collecting new training data for the other:

- Improved reasoning allows for filling in larger holes in informal arguments, allows for translating and interpreting inputs specified more informally.
- Improved informal to formal translation expands the amount of data to guide mathematical exploration.

4 Potential Implications of Successful Autoformalization

Autoformalization is not just a challenge: successful autoformalization would represent a breakthrough for general AI with significant implications in various domains.

Autoformalization would demonstrate that sophisticated natural language understanding between humans and AI is feasible: machines could communicate in natural language over ambiguous content and use it to express or guide internal experiences. It would serve as a clear demonstration that natural language is a feasible communication medium for computers as well.

By nature, autoformalization would have immediate practical implications for mathematics. Initially, a strong autoformalization system could be used for verifying existing and new mathematical papers and would enable strong semantic search engines.

In the more general sense of formalization, a solution to autoformalization could give rise to programming agents that turn natural language descriptions into programs. Since programming languages can be formalized completely, reasoning systems trained on mathematical formalization could be fine-tuned for the task of creating algorithms in specific programming languages. By formalizing domain level knowledge, the system could learn to produce code from natural language input. Such reasoning systems should be able to create the formal specification of the task, the executable code and correctness proof of the newly designed algorithm, all at the same time.

Furthermore, this would give rise to strong and flexible general purpose reasoning engines that could be integrated into AI applications, combining reasoning with perception. This could be used to infuse strong reasoning capabilities into other AI systems and serve as a basis for a wide range of such applications (for example semantic search, software synthesis and verification, computer aided design, etc.).

5 Hurdles of Autoformalization

Designing and implementing a strong autoformalization system is a difficult undertaking and is subject to several current research efforts. Let us start with the outline of some straighforward attempt at its construction and analyze its most likely failure modes. We assume a system based on the following two components:

1. A reasoning engine (theorem prover),
2. a translation component for translating informal (natural language) statements into formal statements.

The translation component could generate multiple formal candidate statements in the context of the previously formalized statements and definitions. The system is successful if it creates formal translations for a substantial fraction of the informal statements after a reasonable number of attempts. (The automated verification of the correctness of translation remains a fuzzy, practical question that is subject to ongoing research, however.)

The first problem with mathematical formalization is that it requires at least some initial core formalization data-sets with a significant amount of parallel corpus in formalized and informal mathematical content. One limiting factor is the cost and effort of generating this seed corpus of translations.

Once we have a somewhat working "seed" translation model, one can try bootstrapping and training the system by generating several candidate translations of mathematical statements and trying to prove/refute each of them until we find a formalization that is correct, but not trivial. This means that we can see at least four major potential failure modes:

1. The seed formalization system is too weak to initiate a feedback loop that can open-endedly improve itself.
2. The system might start to generate mistranslations for further training of the translation model, entering a feedback loop of increasingly worse translations.
3. Translation gets stuck: it would generate a lot of incorrect statements that are never verified; and the system stops to improve.

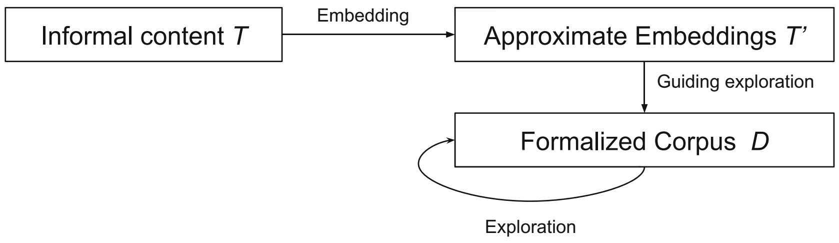
4. The translation never crosses domain boundaries: it formalizes significant parts of some domain, but never succeeds in generalizing to new domains; and the training gets stuck after formalizing limited parts of the corpus.

Natural language is context dependent: it might contain hidden assumptions in far away parts of the text which are impossible to find without a thorough search. For example, papers tend to refer to text books for "basic terminology". The formalization system would need to look up the textbook and mine it for all the subtleties of its definitions and verify that those definitions are consistent with those in the repository of the formal system. If not, then the system would need to create new formal definitions that match those in the paper. Moreover, the paper itself might use inconsistent notations: "abuses of language". Additionally, it might just have plain errors obvious to the human reader. Therefore a straightforward translation attempt is not a robust solution and is unlikely to work in practice.

6 A Proposed Path to Autoformalization

It is hard to anticipate solutions of the potential problems from the previous section. Still, one can aim at designing a system that has a plausible chance of being bootstrapped without getting stuck or misguided.

Instead of direct translation, we propose to rely on a combination of exploration and approximate translation. By "approximate translation", we mean that the translation model does not produce concrete formal transcriptions but approximate embedding vectors thereof. These are then used as guides for an exploration algorithm as the following diagram shows:



The first technical issue concerns the input format of informal mathematical content (i.e. mathematical papers and text books). A textual representation could work well for use cases that do not require the understanding of formulas, diagrams and graphs. However, mathematical content often uses a lot of formulas and diagrams. Geometric illustrations play a role in informing the reader as well. The safest path seems to rely on images instead of textual representation. While this puts more burden on the machine learning part of the system, it can reduce the engineering effort significantly.

Let S denote the set of syntactically correct formal mathematical statements in some formalization environment (e.g. HOL Light). By S', we denote those statements with a formal proof already present in our database. C is the set of possible forward reasoning rules called "conversions". C consists of partial functions with signature $c : S'^{*} \longrightarrow S$, that given a sequence s of true statements in S' it either generates a new true statement $c(s)$ or it just fails. Our system will rely on a few deep learning models:

- An embedding model $e_\theta : S \longrightarrow \mathbb{R}^n$ that embeds formal mathematical statements as low dimensional vectors.
- An approximate translation model $a_\xi : \mathbb{R}^{k \times l} \longrightarrow \mathbb{R}^n$, that outputs an approximate embedding of the formal translation of the informal input statement (given as a picture).
- An exploration guidance model $g_\eta : S \times \mathbb{R}^n \times \mathbb{R}^n \longrightarrow [0,1]^C \times [0,1]$. This model acts as a premise selection model, combined with a conversion type prediction, assuming a finite number of possible conversions. $f_\eta(s,t,p)$ which takes a formal statement s, a target embedding t and the embedding for additional conversion parameters and predicts probabilities for the best conversion steps $[0,1]^C$ and conversion parameter list p at the same time. The exact working of such models is described in [4].

For technical simplicity, we made the simplifying assumption that statements and input images are represented by fixed dimensional vectors, but this is not essential and could be easily changed. The parameters of the deep learning models e_θ, a_ξ and g_η are trained in lock step as described below.

The system is designed to learn to explore the set of true statements while this exploration is guided by a set of target embeddings. These target embedding vectors are produced by translation model a_ξ. During the process, we maintain the following data sets:

- A fixed set of informal target content $T \subseteq \mathbb{R}^{k \times l}$ in raw image format. (The pages containing the statements and definitions we are aiming to formalize.)
- The image of T'_ξ of T under the approximate translation model: $\{a_\xi(t) | t \in T\} \subseteq \mathbb{R}^n$. (The predicted embeddings of the informal statements on the formal side.)
- A set of already explored mathematical statements $D \subseteq S'$ of true and proven statements,
- The embeddings of the explored mathematical statements: $D'_\theta = \{e_\theta(s) | s \in D\} \subseteq \mathbb{R}^n$.

Our goal is to find a subset of $D' \subseteq D$, whose image under e_θ aligns well with T'. If our translation model is reasonably good, then true – but non-trivally true – translations are likely to correspond to their informal description. We train all the models simultaneously while we update the datasets T', D and D' as we go.

The goal of embedding model e_θ is to map semantically similar statements to vectors that are close. One can train such models in some supervised, end-to-end manner for one or multiple concrete semantic tasks. For example, the model could embed statements in order to predict whether the statement is useful for proving another specified statement, cf. [1] and [4]. Another related semantic task is that of reasoning in latent space [33], in which the model is trained to perform approximate rewrite operations in the embedding space.

For processing the natural language input, our computer vision model a_ξ predicts $a_\xi(p) = e_\theta(t(p))$, where $t(p)$ stands for the hypothesized formalization of page p. Since e_θ is assumed to be an embedding model that reflects semantic similarity, t can be multi-valued, reflecting that there are several correct formal translations of the same informal statement, the embedding vectors of which are expected to cluster in $\mathbb{R}^n$.

In order to create a feedback loop between training θ and ξ, we maintain a set of proved theorems, a large set of informal statements P and translations $T_\xi = \{a_\xi(p)|p \in P\}$ of approximate translations of informal statements. To generate the training data for training, we run guided exploration by sampling forward reasoning steps using another deep neural network g_η starting from our already proved theorems with the goal of getting close to as many of the approximate translated embeddings T_ξ as possible. For this purpose, η is trained via reinforcement learning in which the reward is based on the negative minimum distance to the closest target embedding vector. The guidance model g_η samples both conversions and conversion parameters ("premises" used for the conversion). Note that g_η can be trained while circumventing the sparse reward problem: even if we do not get close to any of our original targets, we can pretend that the embedding of the statement we arrived at was our original goal from the start. This idea is known as hindsight experience replay [2].

Once our guided search finds enough statements that match some of the prescribed embeddings in T_ξ, we would check that they are non-trivially true and use them as verified translations for retraining a_ξ. As we proceed, we can incrementally train e_θ and g_η as well. For example e_θ could be trained by analyzing the dependency structure of the explored statements (the tactic parameters that led to the new statement), while g_η is trained using reinforcement learning utilizing the rewards collected during exploration.

The main advantage is that this system is expected to be more robust to errors and incomplete inputs: if exploration is powerful enough, then it can work even if we fail to translate some of the statements properly. Also, if formalization gets stuck, the system can just relax the distance with which it accepts formalization attempts in the embedding space, still producing valid theories that might not exactly correspond to the informal corpus.

Also the system should be able to generalize to completely new domains more easily, as exploration is more likely to be efficient in the early stages. This can bootstrap the easy parts of the system and can prime the translation model and later exploration so that it can continue bootstrapping successfully.

7 Further Ideas and Considerations

The previous section has given a rough outline of a system that could bootstrap itself for mathematical reasoning via autoformalization. Here we consider additional details that are less critical but helpful for engineering a system described in Sect. 6.

7.1 Choice of Foundation and Framework

Traditionally, many people would argue that the choice of the right framework and foundation is crucial for the success of a formalization project. For human users of interactive proof assistants the right framework can affect the productivity of formalization, but generally these effects are hard to quantify and there had been several types of logical foundations and frameworks that have been applied successfully in large scale formalization efforts: Mizar [38], HOL Light [23], HOL4 [45], Isabelle [53], Coq [7], Metamath [37] and Lean [8]. We have only listed proof assistants that have demonstrated a significant amount of successful formalization efforts: tens of thousands of theorems, some of them of great complexity.

A few considerations apply when it comes to automatically formalizing from natural language. Theorem libraries based on purely constructive non-classical logic might have a significant mismatch with most mainstream mathematical text for non-constructive mathematical objects. Also it is useful if the proof assistant can be extended easily with new high level algorithms (new "tactics", for example). Engineering efforts required to interface with external libraries, especially machine learning systems is a point of consideration, too.

One last concern is the expressiveness of the logic. Although first order logic is generally capable of expressing the Zermelo-Fraenkel axiom system, it requires maintaining axiom schemes which is a handicap. Based on these considerations, higher order logic based systems with large theorem libraries (HOL, Isabelle, Coq, Lean) seem to be best suited to be the foundation of an autoformalization system.

7.2 Unsupervised Pretraining Tasks

Self-supervised pretraining might become an enabling factor for autoformalization system. BERT [9] style pretraining for both formal and informal corpora can pave the way, but formal content allows for much more creativity and possibility for pretraining models, these include training "skip-tree" models that generate missing trees from their context. This task subsumes a lot of other logical reasoning tasks

1. Skip-tree: removing some random or strategically selected subtree and predict the whole missing subtree.
2. Type inference model: Learn to do (partial) type inference of formulas.
3. Predicting the embedding or the text of possible useful lemmas that could help proving the statement.
4. Predicting the result (embeddings) of rewrites.
5. Predicting substitutions or inductive invariants.
6. Given a subtree, predict the containing tree.
7. Rewrite a formula with a sequence of rewrites, try to predict the sequence of rewrites that has lead to the result.

Prior work also includes predicting the symbolic integral of expressions [31] and predicting how general mathematical statements behave under rewrites in the latent space [33].

7.3 Additional Technical Considerations

For the neural representation of formal content, the network architecture has a significant effect on the performance of the reasoning system. Currently, deep graph embedding networks [40,52] with node-sharing do best, however transformer networks [51] have yielded breakthrough on formal integration [31], recently.

Our main approach is based on forward exploration. Aligning the result of forward exploration with the target statement might require reverse (goal oriented) proof search, however. As most research is done on reverse proof search e.g. [4,26,55], integrating with such methods is likely a useful idea and a fruitful engineering direction.

As described in the Sect. 6, we need to filter out translation candidates that are incorrect, trivial or uninteresting. The first criterion is clear: we do not expect wrong statements to be correct formalization candidates. It is harder to discard candidate translations that are trivially true (e.g. due to too general assumptions or other translation errors). This could be identified by observing the hardness of proving statements. Also, if a statement is overly long, or has a lot of redundant subtrees, then it is highly unlikely to come from formalizing human content. The usefulness of the produced statements should give another strong indication of good translations.

Curriculum learning is a promising way of learning to find longer proofs. A remarkable result demonstrating the power of strong curriculum is [61] in which they trained a reinforcement learning system to find proofs consisting of several thousand elementary proof-steps without any search, just by letting the policy network predict them in a single run.

Tactics in proof assistants are subroutines that perform complicated algorithms in order to produce long chains of arguments about the correctness of certain formulas. Examples of existing tactics include the application of SAT-solvers or first order automated provers to prove statements that require simple logical reasoning, but they can be as complex as using Gröbner bases of ILP solvers to reason about polynomial equations or linear systems of Diophantine inequalities. Given the complexity of such algorithms, it is unlikely that one could synthesize a general purpose computer algebra system from scratch initially. However, the vast majority of sophisticated human mathematics was discovered without the aid of computer programs, so we can hope that matching the performance of human mathematicians could be achieved without synthesizing complicated tactics.

For refutation and counterexample generation, it might be important to find substitutions into statements that provide a refutation of that statement. In general it is a promising research direction to use deep learning based models to embed not just the syntactic form of formulas, but also some experience stream associated with experimentation with the statements.

One difference between theorem proving and game playing engines is the much wider breadth of mathematics. For neural network based systems, this might mean that it could require very large neural networks to distill all the skills required to cope with all areas of mathematics at once. One could try to cope with that by utilizing mixture of expert models [58]. However, their fixed gating mechanism and rigid model architectures are relatively hard to extend. More flexible are multi-agent architectures using artificial market mechanisms that allow arbitrary agents to bet on the status of mathematical conjectures while the agents are rewarded for correct predictions, proving theorems formally and for introducing interesting new conjectures. The above direction opens a large box of interesting mechanism design [39] questions. [12] proposes that a betting market based multi-agent system under resource constraints is useful for assigning consistent probability values to mathematical statements. This could give some theoretical backing and guidance towards such solutions.

8 Short History of Autoformalization

The idea of autoformalization was first presented in 1961 by John McCarthy [36]. Another early attempt was the 1990 doctoral thesis of Donald Simons [44]. A first thorough study was performed in the 2004 doctoral thesis of Claus Zinn [60]. These works did not result in even partially practical solutions.

Josef Urban started to work on the topic in the early 2000s. He devised a first large scale benchmark for reasoning in large theories [48], motivated by the insight that reasoning in the presence of a large knowledge base of mathematical facts is a critical component in any autoformalization system. In 2007, he published the pioneering MaLARea [50] system for reasoning in large theories. From then on, with Cezary Kaliszyk they have been spearheading the research on reasoning in large theories and autoformalization [28,29].

9 Indications of Feasibility

Given the great complexity and breadth of the problem, it is justified to ask why is autoformalization even considered as a realistic goal in the short term – that is, within years. This section tries to give heuristic arguments for the feasibility of this task by methods that are either known or are on a clear improvement trajectory.

The success of autoformalization hinges on solving two difficult-looking tasks:

1. General purpose symbolic reasoning
2. Strong natural language understanding

The thesis of this paper is that deep learning will enable the advancement of both of those areas to the extent that is necessary for human level formalization and reasoning performance in the coming years. Let us review their recent progress in separation with the focus of exploring how they could enable autoformalization.

9.1 Search and Reasoning

Recently, it has been demonstrated by AlphaZero [42] that the same relatively simple algorithm based on Monte Carlo tree search (MCTS) [30] and residual convolutional networks [24] could achieve higher than human performance in several two-person games: go, chess and shogi, by self-play alone, utilizing the same algorithm for each of those games, **without** learning on any human expert games at all. Effectively, AlphaZero was able to rediscover all of the important chess, go and shogi knowledge in a few days that took human players centuries to discover.

However, mathematical reasoning differs from game playing in many respects:

1. The impossibility of self play: If open ended exploration is considered as an alternative, it is to decide what to explore. The lack of self play makes automated curriculum learning much harder for theorem proving.
2. Large, indefinitely growing knowledge base, resulting in a virtually infinite action space.
3. Very sparse reward: In the case of theorem proving, it is very hard to assign reward to failed proof attempts.
4. The diversity of mathematical knowledge: by nature, two-player games are very coherent, since each player has to be able to answer any move by any other player. Mathematics consists of wide range of loosely connected disciplines and it takes a lot of human experts to cover each of them.

DeepMath was the first attempt for applying deep learning at premise selection for the Mizar corpus [49] via convolutional networks and it yielded some initial improvements for this task. Also theorem prover E was improved by integrating neural network guidance [35]. In 2017, TacticToe [14], has demonstrated that tactic based higher order theorem proving via machine learning (even without the use of deep learning) is possible.

More recently the DeepHOL system [4] gave further demonstration of the power of deep learning in the more general case: for higher order logic and in the presence of a large knowledge base of premises to be used. However, formulas can be best described as graphs, suggesting the use of graph neural networks, which was suggested first in [52] and then yielded significant gains (40% relative increase in success rate) on the HOList benchmark in the end-to-end proving scenario [40]. DeepHOL-Zero [3] has demonstrated that relatively simple exploration heuristic allows for bootstrapping systems that can learn to prove without existing human proof logs to train on. While the proofs created by the above systems are very short, [61] demonstrates successfully, that with the right curriculum, in their limited setting, it is possible to train models that create proofs of several thousand steps without error.

9.2 Natural Language Processing and Understanding

Since 2017, natural language processing went through a revolution similar to that of computer vision, due to new neural model architectures, especially transformer networks [51] and large scale self-supervised training on vast corpora [9,41,56]. This has spurred fast advances in machine translation and language understanding. On some of the benchmark, this has resulted in human or close to human performance, for example on SQuAD 1.0 [57]. However this has lead to development of improved benchmarks to target the common weak points of those algorithms. Progress is still strong in this domain: improved model architectures and better tasks on larger corpora have yielded significant gains at a steady pace. On the analogy with computer vision, one can also foresee that natural architecture search will give rise to further advances in this field as well. Autoformalization systems can leverage all those advances for stronger translation models from natural language to the embedding space of formal statements.

9.3 Overview

Here is a short overview of the factors that support the potential success of autoformalization in the coming years:

1. The success of deep learning infused search in two person games, especially AlphaZero [42] style Monte Carlo tree search [30].
2. The demonstrations of the usefulness of deep learning in automated reasoning: premise selection [1] and proof guidance [4,35,40]
3. The demonstration that automated proof search can be learned without imitation [3].
4. The fast progress and success of neural architectures for formal and natural language content, especially graph neural networks [40,52,54] and transformers [51] for symbolic mathematics [31].
5. The success of imposing cyclic translation consistency [59] in image generation and unsupervised translation [32] give strong indications that autoformalization could be bootstrapped using very limited set of labeled pairs of formalized theorems.
6. The success of hindsight experience replay [2] to address the sparse reward problem for robotics applications.
7. The quick pace of progress in natural language processing via large, deep network models, and large scale self-supervised pretraining. Impressive results in several translation and natural language understanding benchmarks [34].
8. Generative neural models improve at a fast pace and yield impressive result in a wide range of domains from image generation to drug discovery.
9. Multi-agent system with agents specialized in different domains [12] could give a rise to open-ended self-improvement.
10. Automated optimization of neural architectures via neural architecture search [47,62] and other automated methods [19].
11. Computational resources available for deep learning purposes are still expanding quickly and are getting cheaper. For example, as of July 2019, Google' TPUv3 based pods can deliver over 100 petaFLOPS performance for deep learning purposes [18].

10 General Summary and Conclusions

We have argued in this paper that:

1. Autoformalization could enable the development of a human level mathematical reasoning engine in the next decade.
2. The implementation of autoformalization presents significant technical and engineering challenges.
3. Successful implementation of mathematical reasoning (theorem proving) and autoformalization has many implications that go far beyond just transforming mathematics itself and could result in the creation of a general purpose reasoning module to be used in other AI systems.
4. A reasoning system based purely on self-driven exploration for reasoning without informal communication capabilities would be hard to evaluate and use.
5. It is easier to engineer and bootstrap a system that learns to perform both formalization and reasoning than either task in separation.
6. It seems easier to create a formalization system from image than text data.
7. A naïve, direct translation approach for autoformalization would be brittle, hard to engineer and unlikely to work.
8. Combining approximate formalization (predicting embedding vectors instead of formulas) and guided exploration is a more promising direction to autoformalization than direct translation.
9. Deep Learning should be crucial for open ended improvement and reaching human level reasoning and formalization performance.
10. Recent progress in neural architectures, language modelling, self- and semi-supervised training, reinforcement learning, automated neural architecture search, and AI driven theorem proving paves the way for strong automated reasoning and formalization systems.

Acknowledgements. My warmest thanks go to my close collaborators and colleagues Sarah M. Loos, Markus N. Rabe, Kshitij Bansal, Francois Chollet, Alex Alemi, Stewart Wilcox, Niklas Een, Geoffrey Irving, Victor Toman and Aditya Paliwal for their contributions towards the goals sketched here. I am also indebted to Josef Urban and Cezary Kaliszyk for their pioneering work and selflessly sharing their vision and expertise and also for their collaboration on this area. I am also thankful to Ilya Sutskever, Henryk Michalewski, Daniel Huang, Quoc Le, Dániel Varga, Zsolt Zombori, Adrián Csiszárik for their feedback and valuable discussions on this topic. I would like to thank to Jay Yagnik, Rahul Sukthankar, Ashok Popat, Rif Saurous, Jeff Dean and Geoffrey Hinton for their support of deep learning based reasoning work at Google. I am grateful to Péter Szoldán, Christoph Benzmüller and Bruce Miller for proofreading the manuscript.

References

1. Alemi, A.A., Chollet, F., Eén, N., Irving, G., Szegedy, C., Urban, J.: Deepmath - deep sequence models for premise selection. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, Barcelona, Spain, 5–10 December 2016, pp. 2235–2243 (2016)
2. Andrychowicz, M., et al.: Hindsight experience replay. In: Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 5048–5058 (2017)
3. Bansal, K., Loos, S.M., Rabe, M.N., Szegedy, C.: Learning to reason in large theories without imitation. arXiv preprint arXiv:1905.10501 (2019)
4. Bansal, K., Loos, S.M., Rabe, M.N., Szegedy, C., Wilcox, S.: HOList: an environment for machine learning of higher-order theorem proving. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Proceedings of Machine Learning Research, Long Beach, California, USA, 9–15 June 2019, vol. 97, pp. 454–463. PMLR (2019)
5. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In: Cleaveland, W.R. (ed.) TACAS 1999. LNCS, vol. 1579, pp. 193–207. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49059-0_14
6. Blanchette, J.C., Kaliszyk, C., Paulson, L.C., Urban, J.: Hammering towards QED. J. Formalized Reasoning **9**(1), 101–148 (2016)
7. The Coq Proof Assistant. http://coq.inria.fr
8. de Moura, L., Kong, S., Avigad, J., van Doorn, F., von Raumer, J.: The lean theorem prover (System Description). In: Felty, A.P., Middeldorp, A. (eds.) CADE 2015. LNCS (LNAI), vol. 9195, pp. 378–388. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21401-6_26
9. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Long and Short Papers), vol. 1, pp. 4171–4186 (2019)
10. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24605-3_37
11. Fitting, M.: First-order Logic and Automated Theorem Proving. Springer, New York (2012). https://doi.org/10.1007/978-1-4612-2360-3
12. Garrabrant, S., Benson-Tilsen, T., Critch, A., Soares, N., Taylor, J.: Logical induction. arXiv preprint arXiv:1609.03543 (2016)
13. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016, pp. 2414–2423. IEEE Computer Society (2016)
14. Gauthier, T., Kaliszyk, C., Urban, J.: TacticToe: learning to reason with HOL4 tactics. In: Eiter, T., Sands, D. (eds.) LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, 7–12 May 2017, EPiC Series in Computing, vol. 46, pp. 125–143. EasyChair (2017)
15. Gawehn, E., Hiss, J.A., Schneider, G.: Deep learning in drug discovery. Mol. Inform. **35**(1), 3–14 (2016)
16. Gonthier, G.: Formal proof-the four-color theorem. Not. AMS **55**(11), 1382–1393 (2008)

17. Gonthier, G., et al.: A machine-checked proof of the odd order theorem. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) ITP 2013. LNCS, vol. 7998, pp. 163–179. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39634-2_14
18. Google's scalable supercomputers for machine learning, Cloud TPU Pods, are now publicly available in beta. https://bit.ly/2YkZh3i
19. Gordon, A., et al.: MorphNet: Fast & simple resource-constrained structure learning of deep networks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, 18–22 June 2018, pp. 1586–1595. IEEE Computer Society (2018)
20. Grace, K., Salvatier, J., Dafoe, A., Zhang, B., Evans, O.: When will AI exceed human performance? evidence from AI experts. J. Artif. Intell. Res. **62**, 729–754 (2018)
21. Hadjeres, G., Pachet, F., Nielsen, F.: DeepBach: a steerable model for Bach chorales generation. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 1362–1371. JMLR (2017)
22. Hales, T., et al.: A formal proof of the Kepler conjecture. In: Forum of Mathematics, Pi, vol. 5. Cambridge University Press (2017)
23. Harrison, J.: HOL light: a tutorial introduction. In: Srivas, M., Camilleri, A. (eds.) FMCAD 1996. LNCS, vol. 1166, pp. 265–269. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0031814
24. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016, pp. 770–778. IEEE Computer Society (2016)
25. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving and verifying the Boolean Pythagorean triples problem via cube-and-conquer. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 228–245. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_15
26. Huang, D., Dhariwal, P., Song, D., Sutskever, I.: GamePad: a learning environment for theorem proving. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019. OpenReview.net (2019)
27. Kaliszyk, C., Urban, J.: HOL (y) hammer: online ATP service for HOL light. Math. Comput. Sci. **9**(1), 5–22 (2015)
28. Kaliszyk, C., Urban, J., Vyskočil, J.: Learning to parse on aligned corpora (Rough Diamond). In: Urban, C., Zhang, X. (eds.) ITP 2015. LNCS, vol. 9236, pp. 227–233. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22102-1_15
29. Kaliszyk, C., Urban, J., Vyskocil, J.: System description: statistical parsing of informalized Mizar formulas. In: Jebelean, T., Negru, V., Petcu, D., Zaharie, D., Ida, T., Watt, S.M., (eds.) 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2017, Timisoara, Romania, 21–24 September 2017, pp. 169–172. IEEE Computer Society (2017)
30. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006). https://doi.org/10.1007/11871842_29
31. Lample, G., Charton, F.: Deep learning for symbolic mathematics. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020. OpenReview.net (2020)
32. Lample, G., Conneau, A., Denoyer, L., Ranzato, M.: Unsupervised machine translation using monolingual corpora only. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018, Conference Track Proceedings. OpenReview.net (2018)

33. Lee, D., Szegedy, C., Rabe, M.N., Loos, S.M., Bansal, K.: Mathematical reasoning in latent space. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020. OpenReview.net (2020)
34. Liu, Y., et al.: RoBERTa: a robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
35. Loos, S., Irving, G., Szegedy, C., Kaliszyk, C.: Deep network guided proof search. In: Eiter, T., Sands, D. (eds.) LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, 7–12 May 2017, EPiC Series in Computing, vol. 46, pp. 85–105. EasyChair (2017)
36. McCarthy, J.: Computer programs for checking mathematical proofs. In: A Paper Presented at the Symposium on Recursive Function Theory, New York, April 1961
37. Megill, N.: Metamath. In: Wiedijk, F. (ed.) The Seventeen Provers of the World. LNCS (LNAI), vol. 3600, pp. 88–95. Springer, Heidelberg (2006). https://doi.org/10.1007/11542384_13
38. The Mizar Mathematical Library. http://mizar.org
39. Nisan, N., et al.: Introduction to mechanism design (for computer scientists). Algorithmic Game Theor. **9**, 209–242 (2007)
40. Paliwal, A., Loos, S., Rabe, M., Bansal, K., Szegedy, C.: Graph representations for higher-order logic and theorem proving. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, 7–12 February 2020. AAAI Press (2020)
41. Peters, M.E., et al.: Deep contextualized word representations. In: Walker, M.A., Ji, H., Stent, A. (eds.) Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, 1–6 June 2018, (Long Papers), vol. 1, pp. 2227–2237. Association for Computational Linguistics (2018)
42. Silver, D., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
43. Silver, D., et al.: Mastering the game of go without human knowledge. Nature **550**(7676), 354 (2017)
44. Simon, D.L.: Checking number theory proofs in natural language. Ph.D thesis (1990)
45. Slind, K., Norrish, M.: A brief overview of HOL4. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 28–32. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71067-7_6
46. Szegedy, C., et al.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, 7–12 June 2015, pp. 1–9. IEEE Computer Society (2015)
47. Tan, M., Le, Q.V.: EfficientNet: rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, California, USA, 9–15 June 2019, Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR (2019)
48. Urban, J.: Translating Mizar for first order theorem provers. In: Asperti, A., Buchberger, B., Davenport, J.H. (eds.) MKM 2003. LNCS, vol. 2594, pp. 203–215. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36469-2_16
49. Urban, J.: MPTP 0.2: design, implementation, and initial experiments. J. Autom. Reasoning **37**(1–2), 21–43 (2006)

50. Urban, J.: MaLARea: a metasystem for automated reasoning in large theories. In: Sutcliffe, G., Urban, J., Schulz, S. (eds.) Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories, Bremen, Germany, 17th July 2007, CEUR Workshop Proceedings, vol. 257. CEUR-WS.org (2007)
51. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017, pp. 5998–6008 (2017)
52. Wang, M., Tang, Y., Wang, J., Deng, J.: Premise selection for theorem proving by deep graph embedding. In: Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 2786–2796 (2017)
53. Wenzel, M., Paulson, L.C., Nipkow, T.: The Isabelle framework. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 33–38. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71067-7_7
54. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. In: IEEE Transactions on Neural Networks and Learning Systems, pp. 1–21 (2020)
55. Yang, K., Deng, J.: Learning to prove theorems via interacting with proof assistants. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, California, USA, 9–15 June 2019, Proceedings of Machine Learning Research, vol. 97, pp. 6984–6994. PMLR (2019)
56. Yang, Z., Dai, Z., Yang, Y., Carbonell, J.G., Salakhutdinov, R., Le, Q.V.: XLNet: generalized autoregressive pretraining for language understanding. In: Wallach, H.M., et al. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Canada, Vancouver, BC, 8–14 December 2019, pp. 5754–5764 (2019)
57. Yu, A.W., et al.: QANet: combining local convolution with global self-attention for reading comprehension. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018, Conference Track Proceedings. OpenReview.net (2018)
58. Yuksel, S.E., Wilson, J.N., Gader, P.D.: Twenty years of mixture of experts. IEEE Trans. Neural Networks Learn. Syst. **23**(8), 1177–1193 (2012)
59. Zhu, J.-Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 2223–2232. IEEE Computer Society (2017)
60. Zinn, C.: Understanding informal mathematical discourse. Ph.D thesis, Institut für Informatik, Universität Erlangen-Nürnberg (2004)
61. Zombori, Z., Csiszárik, A., Michalewski, H., Kaliszyk, C., Urban, J.: Towards finding longer proofs. arXiv preprint arXiv:1905.13100 (2019)
62. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017, Conference Track Proceedings. OpenReview.net (2017)