Left to the Reader: Abstracting Solutions in Mathematical Reasoning

Gabriel Poesia[1] **(poesia@cs.stanford.edu)** and **Noah D. Goodman**[12] **(ngoodman@stanford.edu)**
[1]Department of Computer Science, Stanford University
[2]Department of Psychology, Stanford University

Abstract

Formal mathematical reasoning is unique in its precision: any valid conclusion can be justified by a sequence of base axioms. But human-written proofs or solutions rarely operate at that level. Instead, obvious steps are skipped to provide a simple, lucid argument. This is especially important in an educational setting, where too many details in an example solution, or too few, can confuse a student. What are the key steps for humans in a given formal solution? We investigate several computational hypotheses in the context of equation solving. Specifically, we take a reinforcement learning agent that solves equations using low-level axioms, and propose a series of methods for abstracting its solutions by selecting key steps. We consider methods based on the semantic distance between subsequent steps, based on the steps with the highest uncertainty for the agent, and based on transitions between latent "high-level skills" learned from a large number of agent-produced solutions. In a human evaluation we find that skill-base simplifications were judged most useful. These results suggest new directions for understanding human mathematical reasoning.

Keywords: mathematical reasoning; reinforcement learning; skill discovery; abstraction

What makes a good argument? In essence, an argument must justify the conclusion, given acceptance of the premises. But this seemingly simple criterion turns out to be challenging to assess: a rich body of theory, dating back to Aristotle (Weston, 2018; Bencivenga, 1979), has attempted to characterize good arguments as well as to identify common fallacies and pitfalls. Many intricacies stem from disagreements on what premises are valid and what inferences can be made. However, even among equally *valid* arguments some seem *better* than others.

In formal mathematical reasoning, for instance, all premises and rules of inference can be precisely stated, and the correctness of proofs can even be mechanically verified by computers. However, human mathematical reasoning is rarely formal: even mathematics researchers do not write down full arguments in a formal fashion. Indeed, a formal proof of a mathematical statement is often more than 10 times longer than a corresponding human proof[1]. The reason for this difference lies in the omission or simplification of many concrete details, which we call *abstraction*: human-written arguments might only loosely mention how axioms or theorems are being applied, and many steps are often left implicit.

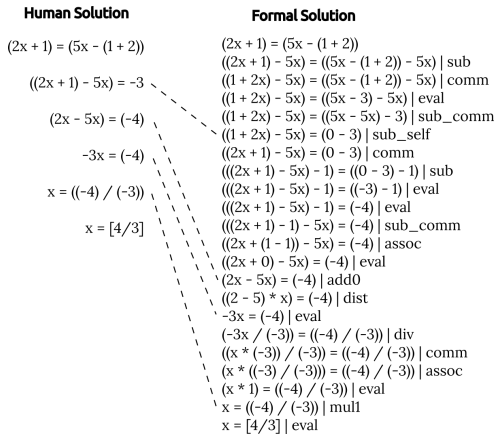

Figure 1: Human solutions for mathematical problems, such as linear equations, are typically much shorter and take more abstract steps than an axiomatic, machine-generated solution. The computational methods we propose attempt to understand and bridge this abstraction gap.

These omissions are sometimes stated in the text: intermediate steps deemed trivial or unimportant are, often frustratingly, "left to the reader".

Abstraction is necessary for the human practice of mathematics. As even Bourbaki, known for a rigorous style of mathematical presentation, concedes:

> "The tiniest proof at the beginning of the Theory of Sets would already require several hundreds of signs for its complete formalization. [F]ormalized mathematics cannot in practice be written down in full". (Bourbaki, 2004)

Thus, in practice, only the most important steps in a proof are written down, for the purpose of conveying to the reader why a given mathematical claim holds. This elicits a natural question: what makes some steps the "most important" in a mathematical derivation? Can we determine them computationally, and gain insight into the cognitive process behind this choice?

The answers to these questions have practical implications for mathematics education. When showing worked examples to a student, providing every axiomatic step would be completely overwhelming. On the other hand leaving out crucial

[1]This is the so-called de Bruijn factor (De Bruijn, 1994), used to assess the conciseness of computer theorem proving languages.

In J. Culbertson, A. Perfors, H. Rabagliati & V. Ramenzoni (Eds.), *Proceedings of the 44th Annual Conference of the Cognitive Science Society.*

steps would leave the example equally opaque. Automated tutoring systems (Ritter, Anderson, Koedinger, & Corbett, 2007; Crow, Luxton-Reilly, & Wuensche, 2018), that aim to help students learn partly by generating examples, would thus benefit from an algorithmic approach to simplifying arguments to their clearest form.

In this paper, we take initial steps toward this goal. We draw from recent work in machine learning models for mathematical reasoning, which provides us with tools to find exhaustive step-by-step solutions to mathematical problems. In particular, the ConPoLe model (Poesia, Dong, & Goodman, 2021) has been shown to learn to solve problems in a set of Common Core educational mathematical domains, such as linear equations and fraction simplification. Given a problem, the trained ConPoLe solver constructs a formal, step-by-step derivation of the solution by applying low-level domain axioms (e.g., commutativity, distributivity, or evaluation of operations with constants). Moreover, it constructs semantic representations of each of these steps, in the form of embedding vectors for each step.

Since ConPoLe operates at the axiomatic level, a solution to even a simple equation can span dozens of steps. Figure 1 shows one example, where the generated solution annotates each step with the domain axiom being invoked (axiom parameters were omitted for brevity). These solutions do not naturally resemble human solutions to the same problems, which tend to be significantly shorter, taking more abstract steps. How can we computationally characterize and close this gap?

We evaluate several alternative approaches to this problem. We start from a set of complete, axiomatic, solutions, and produce simplified solutions by taking a subset of the original steps, based on alternative notions of step importance. Specifically, we evaluate (i) selecting solution steps that make the largest changes to the equations (as measured by ConPoLe's representations), (ii) selecting solution steps that are the least predictable (as measured by ConPoLe's step uncertainty), and (iii) selecting solution steps at the boundaries between latent higher-level "skills" (recovered by an unsupervised skill-discovery method). We compare a simplified solution to a control alternative that chooses (a matched number of) random steps from the original solution. We find that human participants prefer the skill-based segmentation most strongly, suggesting that considering hierarchy and strategy is important to constructing human-like solutions. This result indicates that artificially-generated solutions can be practically useful for education, and theoretically useful to understand human mathematical abstractions.

Domain and Solver

We take the Common Core Equations environment and the ConPoLe model from Poesia et al., 2021. This environment can generate random linear equations from syntactic templates taken from the Cognitive Tutor Algebra dataset (Ritter et al., 2007). Given an equation, the environment can then enumerate allowed actions and the resulting equation of applying them, using a set of primitive mathematical axioms (such as commutativity of addition and multiplication, or adding a constant to both sides of the equation).

ConPoLe was trained to find solutions in this environment: starting at an equation s, its learned policy π defines a probability distribution $\pi(s_{t+1}|s_t)$ over the set of next states $A(s_t)$ allowed by the environment; choosing an action leads to a new state. The solution is complete when the environment detects that a solution has been found (i.e., the equation has the form "x = c" for a constant c).

The ConPoLe policy is defined by taking small steps in a learned state representation space, $\phi(s) \in \mathbb{R}^{512}$:

$$\pi(s_{t+1}|s_t) \propto \phi(s_t)^\top \cdot M_\theta \cdot \phi(s_{t+1})$$

We note that because ϕ is trained so that this small-step policy can find solutions to equations, distance in this embedding is a good proxy for "conceptual distance" between equations (see Poesia et al. (2021) for more discussion).

Abstracting solutions

In this paper we consider abstraction by simplifying solutions: keeping only "key" steps and skipping the rest. How can we computationally define the key steps in a mathematical solution? We consider three possible answers, based on conceptual distance, uncertainty, and high-level skills. Each answer leads to a method for simplifying solutions.

Using distance

One hypothesis is that a step's importance to the reader will depend on how much it changes the state of the solution. For example, in an equation such as $2x = (10 + x) - x$, we can use associativity on the right hand side to arrive at $2x = 10 + (x - x)$. Though this moves toward the solution, the equations before and after this step look rather similar. For the next step, apply the property that subtracting anything from itself yields 0, to obtain $2x = 10 + 0$. This change is intuitively larger than the previous, both in form and meaning. Our distance-based simplification methods keep the intermediate steps right immediately following the largest changes to the equation.

For this use, the measure of "change" should relate to states becoming closer to the solution. Purely syntactic metrics fail to capture this notion. For example, the edit distance from "x + 1 = 2" to both "(x + 1) - 1 = (2 - 1)" and "(x + 1) - 2 = (2 - 2)" is the same, but only the first step moves towards a solution. Thus, we use the Euclidean distance between state representations computed by ConPoLe. Precisely, if s_t and s_{t+1} are two consecutive steps in a solution, we define their distance as $d(s_t, s_{t+1}) = ||\phi(s_t) - \phi(s_{t+1})||_2$.

We tested two methods that simplify a solution based on distances:

Distance: Keep only the intermediate steps s_t such that $d(s_{t-1}, s_t) \geq \Delta$, for a fixed hyper-parameter $\Delta \geq 0$.

NDistance: First, to normalize distances, compute the largest distance between consecutive steps in the solution: $\Delta_{max} = \max_t \{d(s_t, s_{t-1})\}$. Then, keep only the intermediate steps s_t such that $d(s_{t-1}, s_t)\Delta_{max} \geq \rho \cdot \Delta_{max}$, for a fixed hyper-parameter $\rho \in [0, 1]$.

Using $\Delta = 0$ or $\rho = 0$ keeps all intermediate steps and produces the most detailed solution possible; conversely, $\rho = 1$ or $\Delta = \infty$ removes all intermediate steps. Intermediate values allow tuning conciseness. Note that the initial equation and the final state are always part of the solution; all methods we describe only filter intermediate states.

Using the solver's uncertainty

Another possible strategy for defining "key steps" in a solution is to base the decision on uncertainty about the next step. Intuitively, the more "obvious" it is what the next step should be, the less important it is to include in the solution. This leads to a simplification method that removes steps to which the ConPoLe solver assigns a very high probability:

Uncertainty: Keep only the intermediate steps s_t such that $\pi(s_t|s_{t-1}) < \rho$, for a fixed hyper-parameter $\rho \in [0, 1]$.

Again, with $\rho = 0$, all intermediate steps are discarded; with $\rho = 1$, all are kept. In between, we find varying abstraction levels.

Using transitions between high-level skills

Uncertainty-based filtering selects transitions between consecutive steps that are the least predictable, according to the solution model. An alternative hypothesis is that key transitions are between more abstract "skills", that are themselves realized through a sequence of lower-level axioms. For example, in the equation $2x = (15 + 4) + (36 + 27)$, one might want to first "simplify the right-hand side", and after that "isolate x". We can carry out each of these high-level operations through more than one path using the basic axioms. Thus, the solution is unpredictable in the immediate next step, but we'd predict with high confidence to eventually arrive at $2x = c$ (for some c) after a few steps. Thus, long-range predictability does not imply step-wise predictability. This suggests that to simplify a long solution, we should infer these latent high-level actions—skills—and select only the states that immediately follow transitions between skills.

Inferring transitions between high-level skills (also called "options," Kipf et al. (2019); Barreto et al. (2019)) is a problem that has received significant attention from the Reinforcement Learning community in recent years. Various methods have been proposed that assume different levels of supervision. We use CompILE (Kipf et al., 2019), which requires a set of solutions but no other supervision. Given a data set of ConPoLe solutions, CompILE fits a latent variable model that jointly (i) partitions solutions into sub-segments and (ii) assigns a high-level skill label in each segment.

CompILE is trained by optimizing a variational approximation to the log-likelihood. Given the dataset of solutions $D = \{s_1, \ldots, s_N\}$, where each solution s_i is a sequence of state/action pairs $s_{i,1}, \ldots, s_{i,K}$, CompILE fits p_θ to maximize the likelihood of D:

$$\theta^* = \max_\theta \log L(D|\theta) := \mathbb{E}_{s \in D} p_\theta(s)$$

The likelihood of a given solution $p_\theta(s)$ is defined in terms of the boundaries between skill segments, the latent skill applied in each segment, and the likelihood attributed by each skill's model to the state/action sequence inside each segment. If b_i is the position of the i-th segment boundary, and z_i is the discrete latent skill assigned to the i-th segment, then CompILE defines the likelihood of θ by marginalizing over b and z:

$$p_\theta(s) = \sum_{b_i} \sum_{z_i} p_\theta(s|b_i, z_i) p(b_i|b_{i-1}) p(z_i)$$

Where $p(b_i|b_{i-1})$ is a Poisson prior over where skill boundaries are placed, and $p(z_i)$ is a uniform prior over what skills are applied. Inside each skill segment, $p_\theta(s|b_i, z_i)$, modeled with a Recurrent Neural Network, assigns a likelihood for the action sequence in that segment given its boundaries and a choice of latent skill.

We refer to the original work for details about model training. Of note for us, the variational inference method used by CompILE also learns a *boundary prediction network*, which takes a solution and outputs the most likely skill boundaries. Thus, after CompILE is trained, we can use it to segment any ConPoLe solution. This yields our last simplification method:

Skills: Infer high-level skill segments in the solution's axiom sequence, then select the states that immediately follow each skill boundary.

Note that skills are learned from sequences of axioms, not their specific parameters (e.g. subtract something from both sides, without regard to what value to subtract). This already introduces some level of abstraction and simplifies the input to CompILE. The rate parameter λ used in the Poisson prior of $p(b_i|b_{i-1})$ controls the expected number of segments inferred in a given solution. This allows us to tune the abstraction level of simplified solutions, in a similar fashion to the other methods we described.

Comparing simplification methods

We evaluate these simplification methods with both offline analyses and a human evaluation. We start by asking whether the simplification methods yield different notions of "important steps" in practice. Can we find distinctive patterns in the steps they select?

Figure 2 shows two measures of the average difference between simplifications produced by each pair of methods. On Figure 2(a), the difference between two simplifications is measured by Jaccard distance: we treat the boundaries placed by each method in the same solution as two sets B_1 and B_2, and compute $D_J(B_1, B_2) = 1 - |B_1 \cap B_2| / |B_1 \cup B_2|$. Since Jaccard is a very strict measure, in Figure 2(b), we use a softer

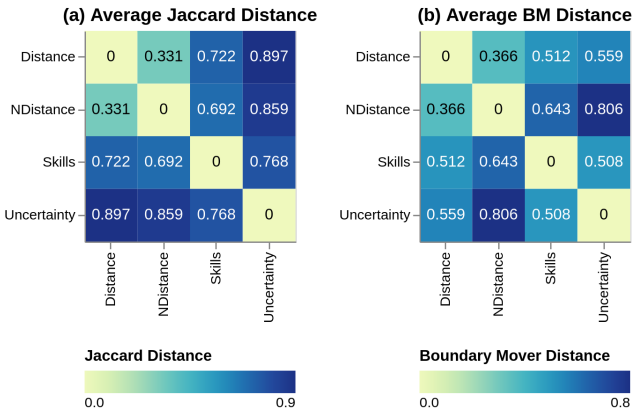

Figure 2: Average distance between simplified solutions for each pair of simplification methods, as measured by (a) Jaccard Distance and (b) the Boundary Mover Distance (an adaptation of string edit distance). In both, all methods have significant disagreements; the largest similarity being between the two distance-based methods (Distance and NDistance).

metric that is similar to the Edit Distance between strings: we compute how many operations are necessary to transform one set of boundaries into another, where atomic operations are duplicating a boundary and moving boundaries by one step, normalized by the maximum possible number of operations given the solution's lengths. Both metrics indicate that all 4 methods have some level of agreement, but also significant differences.

We further explore their differences in two analyses. Figure 3 shows the distribution of the average length of each of these methods in solutions of varying lengths. Importantly, all methods were calibrated to have a median compression of 50%. Yet we observe that they distribute abstraction levels unevenly, and differently from each other.

Finally, Table 1 shows a kind of "fingerprint" of the behavior for each method: the most common axioms surrounding the states selected to be kept. Each pattern is an axiom trigram: one axiom used to derive the state right before the selected state, the axiom that derived the selected state, and the axiom that then followed in the original solution. Distance and NDistance select states very frequently surrounded by a small number of axiom patterns. Uncertainty is significantly more diverse than the distance-based methods, and Skills even more so. Altogether, these analyses show that the methods provide distinct characterizations of the importance of solution steps.

Evaluating Human Preferences

We next set out to determine whether the above solution simplification methods yield good solutions according to human judges.

Materials We generated 100 equations with solutions using the Common Core Equations environment. We then applied

Table 1: Axiom trigram patterns around the states selected by each simplification method. In each trigram, the axiom in the middle was used to derive a state in a simplified solution; the frequency in parenthesis shows the fraction of selected states across all solutions that are surrounded by that pattern. The axioms shown here are `eval` (evaluate an operation with constants), `mul1` (remove a multiplication by one), `add0` (remove an addition of zero), `assoc` (associativity), `div` and `sub` (divide or subtract a term on both sides of the equation). Distance and NDistance capture the same 3 patterns, with high concentration on the most frequent trigrams. Skills and Uncertainty are more diverse in where they place boundaries, with distinct patterns across methods.

Method	Frequent axioms surrounding selected states
Skills	eval→ mul1→ eval (8.1%) assoc→ eval→ div (2.8%) eval→ add0→ assoc (2.6%)
NDistance	eval→ mul1→ eval (32.8%) eval→ eval→ mul1 (10.7%) assoc→ eval→ mul1 (6.4%)
Distance	eval→ mul1→ eval (20.7%) eval→ eval→ mul1 (6.6%) assoc→ eval→ mul1 (5.5%)
Uncertainty	eval→ assoc→ eval (4.9%) add0→ assoc→ eval (4.4%) sub→ assoc→ eval (4.2%)

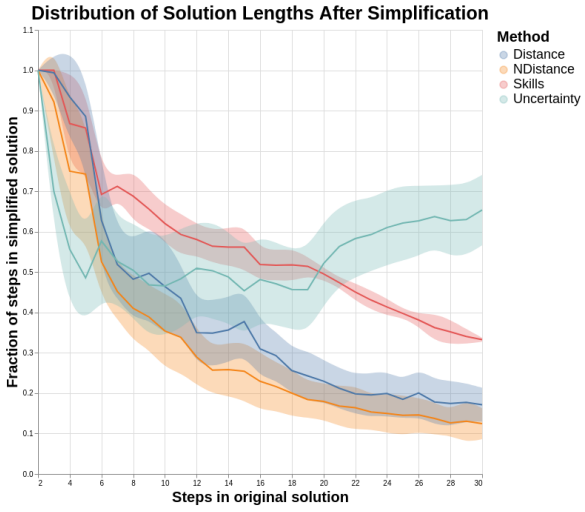

Figure 3: Distribution of solution lengths after simplification for each method. Lines show the mean relative length for each method (simplified length over original length); bands show standard deviation around the mean. Although all methods were tuned to have a median relative length of $\frac{1}{2}$, they produce distinctive distributions of relative lengths.

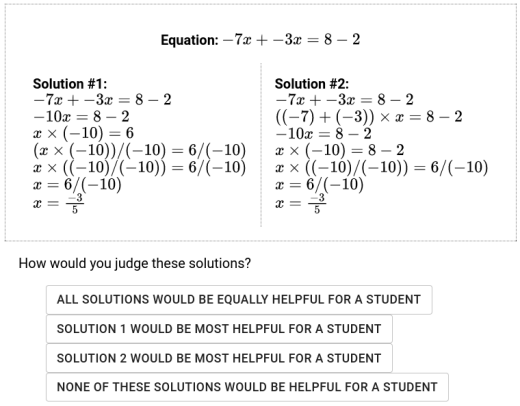

Figure 4: Example of human task. Two solutions to the same equation are shown side-by-side: both are selected steps from a longer formal solution, with matching lengths.

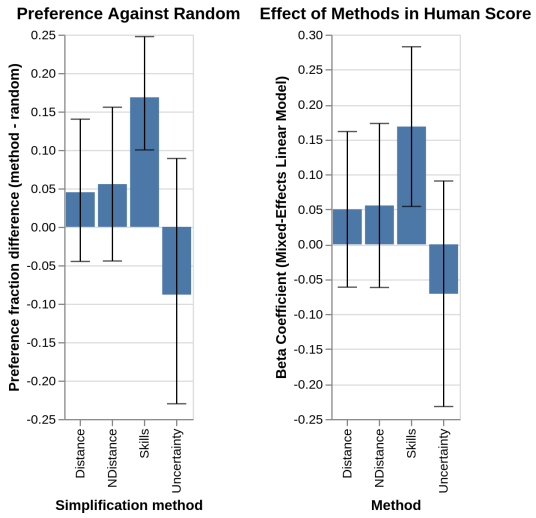

Figure 5: Left: difference between the fraction of human judgements where each method was preferred and those where the random simplification was preferred. Right: fitted Beta coefficients to each simplification method when predicting human scores in a mixed-effects linear regression. Error bars are 95% confidence intervals. Both analyses show Skills as significantly better than random simplification, unlike other methods.

each of the four simplification methods above to all equations, obtaining four simplified solutions for each equation. Asking humans to directly compare different simplifications to one another is challenging because the number of steps selected often differs between methods for any given problem, and yields a strong visual confound. Therefore, we instead chose to evaluate the quality of solutions compared to a random control that matches the simplification length. In other words, for each problem p_i, with a corresponding formal solution s_i, and each simplification method m_j, $m_j(s_i)$ is a simplification of s_i that starts with p, ends in a solution state, and is a subset of s_i; we produce a random simplification $m_j^{rand}(s_i)$ for comparison that has the same first and last states, and selects the same number of intermediate steps from s_i uniformly at random.

Participants For each simplification method, we recruited 18 participants from Amazon Mechanical Turk, that each evaluated 10 pairs of solutions selected randomly from the original set of 100 equations. Thus, we had a total of 72 participants, producing 720 evaluations (180 for each method).

Methods Participants were told that they were helping to improve automated tutors: "These computer tutors can help students learn to solve equations, fractions or other problems. But the computer often finds complicated solutions that are hard for students to understand! Your task is to compare different solutions generated by a program and choose the one you think would be most clear and helpful for students." In each evaluation (Figure 4), we showed a problem (a linear equation) and two simplified solutions, one produced by one of the described methods and one produced by random simplification matched in length. Their order on the screen was randomized. Given a pair of solutions, participants were asked to judge them based on their helpfulness for an algebra student. They could choose from 4 options: "Solution 1 [or 2] would be most helpful for a student", or, in case they were thought to be equivalent, either "All solutions would be equally helpful for a student" or "None of the solutions would be helpful for a student".

Results

Figure 5 shows the results of two convergent analyses of the results. Figure 5 (left) shows the difference between the fraction of responses where each method's simplification was chosen and the fraction in which the random simplification was chosen instead. A positive number thus means that the method's simplification was chosen more often. In this analysis, judgements where both solutions were considered equivalent (either "both would be equally helpful" or "none would be helpful") are ignored. Figure 5 (right) shows an analysis considering all data points: we model participants as random effects and the method as a fixed effect in a mixed-effects linear regression, where we predict a score that summarizes the human judgement (1 when the method's solution was preferred, -1 when the random simplification was preferred, and 0 otherwise). We model $s \sim$ `M + (1 | P)` with s being the vector of human scores, M being a fixed slope for each simplification method and P denoting participants.

We observe that Skills was the most successful method: it was the only method chosen significantly more often than random at the 95% confidence level. Both distance-based methods performed similarly to each other, and Uncertainty's simplifications were chosen less often than the random simplifications (though not significantly). These results suggest that the best simplifications for humans are produced not by

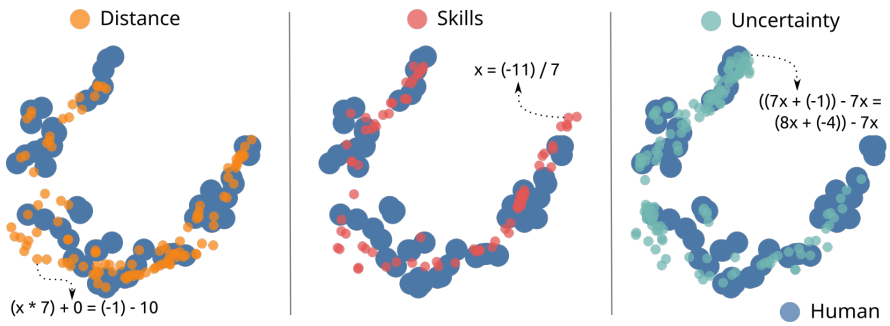

Figure 6: t-SNE visualization of states that were selected exclusively by each simplification method (Distance and NDistance shown aggregated), on top of states that were present in solutions preferred by humans but not in their alternatives. We also identify one state from the region where each method tends to focus more than the others.

analyzing individual transitions (as Uncertainty, Distance and NDistance do), but rather by finding meaningful abstract subsequences (which Skills attempts to do explicitly). Moreover, this result indirectly supports the intuition that human reasoning about mathematical solutions is hierarchical: one might first decide to apply a high-level skill (e.g., "simplify the right-hand side of the equation"), which will necessitate a sequence of smaller steps (e.g., doing a series of calculations with constants); then another (e.g., "isolate the unknown"), and so on. Large changes or uncertainty in the middle of applying the skill might then be less important for conveying the solution than the sequence of skills that is applied.

Visualizing Preferred States

The latent space that ConPoLe uses for equations allows us to further explore (i) what states humans tend to prefer, and (ii) what states each of the methods tend to select more often. Figure 6 shows a t-SNE (Van der Maaten & Hinton, 2008) projection that illustrates these preferences both of methods and of human evaluators. Here, each point corresponds to an equation – either one of the starting equations generated for human evaluation or one of the intermediate states in the complete solutions. To understand which states humans prefer to be included in solutions, we show the states that were present in simplified solutions that were chosen by human evaluators but not in their alternatives (blue points, reproduced in all three plots). Thus, these states are only taken from solutions where all participants agreed on what was the most clear simplification between the two choices[2]. We additionally show the states that were selected exclusively by each method. In other words, we display a state here if any of the given methods was the only to select that state when simplifying the original formal solution.

Skills is the most parsimonious method: only 76 states appear exclusively in simplifications produced by Skills, compared to 114 by the distance-based methods and 142 by Uncertainty. We notice that the ConPoLe latent space is highly structured: when projected to 2 dimensions with either PCA or t-SNE, it reveals a U-shape where solution states are in one end and the longer starting equations cluster on the other end. Compared to other methods, Uncertainty tends to select more states at the beginning (when equations are longer are more actions are available), and fewer states closer to the solution. Distance distinctly selects states at the intermediate stages at a much higher density than other methods. In contrast, Skills spreads states more uniformly (including at the very end, where fewer states were distinctively preferred by humans). Furthermore, we notice gaps between clusters of states preferred by participants, which indicates a hierarchical structure in solution states that the explicit approach of Skills might capture better.

Discussion and Conclusion

Arguments may be good or bad for many reasons. Even among valid mathematical arguments some are better. Of all steps in a complete formal derivation, only a selected few are explicitly written down. In our investigation of computational methods that attempt to reproduce this selection of most important states, a hierarchical approach based on discovering latent skills in the underlying solutions yielded the most clear simplifications, according to a human evaluation.

One aspect that is not captured by our methods is that the most appropriate level of abstraction for communicating a given solution depends on the target reader. For example, undergraduate and middle-school students would likely prefer different levels of detail in a solution to an equation. Understanding how to computationally characterize this preference is an important avenue for future work.

Finally, simplifying by selecting solution steps can be limited if the original solution takes unnatural paths to start with. For example, after subtracting two equal terms, producing 0, ConPoLe often keeps the "+ 0" in the equation for several steps while working on other parts. Humans prefer to simplify early, but that preference does not naturally arise in a learned solver like ConPoLe. How can we characterize these preferences and train models that share them?

[2]Since participants were asked to judge whole solutions, even full agreement might be a noisy indication of state-wise preference.

Acknowledgements

This work was supported by a NSF Expeditions Grant, Award Number (FAIN) 1918771.

References

Barreto, A., Borsa, D., Hou, S., Comanici, G., Aygün, E., Hamel, P., ... others (2019). The option keyboard combining skills in reinforcement learning. In *Proceedings of the 33rd international conference on neural information processing systems* (pp. 13052–13062).

Bencivenga, E. (1979). On good and bad arguments. *Journal of Philosophical Logic*, *8*(1), 247–259.

Bourbaki, N. (2004). Theory of sets. In *Theory of sets* (pp. 65–129). Springer.

Crow, T., Luxton-Reilly, A., & Wuensche, B. (2018). Intelligent tutoring systems for programming education: A systematic review. In *Proceedings of the 20th australasian computing education conference* (p. 53–62). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3160489.3160492 doi: 10.1145/3160489.3160492

De Bruijn, N. G. (1994). A survey of the project automath. In *Studies in logic and the foundations of mathematics* (Vol. 133, pp. 141–161). Elsevier.

Kipf, T., Li, Y., Dai, H., Zambaldi, V., Sanchez-Gonzalez, A., Grefenstette, E., ... Battaglia, P. (2019). Compile: Compositional imitation learning and execution. In *International conference on machine learning* (pp. 3418–3428).

Poesia, G., Dong, W., & Goodman, N. (2021). Contrastive reinforcement learning of symbolic reasoning domains. In A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems.*

Ritter, S., Anderson, J. R., Koedinger, K. R., & Corbett, A. (2007). Cognitive tutor: Applied research in mathematics education. *Psychonomic bulletin & review*, *14*(2), 249–255.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(11).

Weston, A. (2018). *A rulebook for arguments*. Hackett Publishing.