Towards Finding Longer Proofs

Zsolt Zombori
Alfréd Rényi Institute of Mathematics

Adrián Csiszárik
Alfréd Rényi Institute of Mathematics

Henryk Michalewski
University of Warsaw, deepsense.ai

Cezary Kaliszyk
University of Innsbruck and University of Warsaw

Josef Urban
Czech Technical University in Prague

Abstract

We present a reinforcement learning (RL) based guidance system for automated theorem proving geared towards Finding Longer Proofs (FLoP). FLoP focuses on generalizing from short proofs to longer ones of similar structure. To achieve that, FLoP uses state-of-the-art RL approaches that were previously not applied in theorem proving. In particular, we show that curriculum learning significantly outperforms previous learning-based proof guidance on a synthetic dataset of increasingly difficult arithmetic problems.

1 Introduction

In 1958 B. F. Skinner, a pioneer of modern behaviorism, in the article "Teaching Machines" [67] noticed that "in acquiring complex behavior the student must pass through a carefully designed sequence of steps, often of considerable length. Each step must be so small that it can always be taken, yet in taking it the student moves somewhat closer to fully competent behavior". His study extended also to the teaching of arithmetic: "The student is expected to arrive at $9 \cdot 7 = 63$, not by memorizing it as he would memorize a line of poetry, but by putting into practice such principles as that nine times a number is the same as ten times the number minus the number ...". The idea of learning using a curriculum of problems is also widely used in machine learning [16, 8, 55, 59] and in this work we apply curriculum learning to automatic theorem proving focusing on arithmetic.

Our work has the following contributions. (1) We introduce a new theorem proving algorithm FLoP (Section 4) based on reinforcement learning and the connection tableau calculus. FLoP uses a meta-learning variation of the curriculum learning algorithms presented by [55] and [59]. (2) We introduce a synthetic dataset of increasingly difficult arithmetic problems organized as RL environments (Section 5). (3) We use this benchmark to compare (Section 6) the performance of our system with state-of-the-art saturation provers Vampire [46] and E [63] guided by human-designed strategies, and with rlCoP [41] – a recently developed RL-based connection tableau prover. FLoP performs very well in comparison with other provers on the problems, demonstrating its ability to learn how to guide search for longer proofs. Better experimental performance could be achieved only by a portfolio (multi-strategy) mode of a single manually optimized rewriting-based system and only after trying several of its autoconfiguration heuristics.

Our datasets presented in Section 5 seem to be particularly suited for machine learning methods: problems are simple, solutions are long, repetitive and rather predictable for humans. Still, state-of-the-art systems struggle with solving some of the problems – see Section 6 for details.

Knowledge Representation & Reasoning Meets Machine Learning Workshop (KR2ML 2019), Vancouver, Canada.

Other works using machine learning to guide a prover [76, 39, 33, 47, 41, 13] usually deal with large mathematical corpora, while we focus on a fragment of Robinson Arithmetic, which is a limited and simple theory. Our reasons behind this narrower focus: (a) We wanted to create a scalable RL benchmark with emphasis on the length of proofs. (b) A symbolic method based on human-designed sets of hints [77] was previously successfully applied in abstract algebra by [44] to discover long proofs and we wanted to check whether learning of long proofs is feasible using the state-of-the-art ML toolset. (c) We wanted interpretable failure modes. In the case of large mathematical corpora, the interpretation of failures may be a hard task because of multiple failures and the complicated structure of the corpora, requiring specialized domain knowledge both in mathematics and with regard to the inner workings of the proof system.

Our code, datasets and all experiment configuration files are available at `http://bit.ly/code_atpcurr`[1]. Supplementary materials including screencasts with gameplays performed in our environments are available at the project webpage `http://bit.ly/site_atpcurr`.

2 Related work

Machine learning datasets and RL environments involving mathematics and logic. The arithmetic dataset which we introduce in Section 5 is geared towards longer proofs and is structurally much simpler than other theorem proving datasets which we list below. One can think about this suite of RL problems as gridworlds of theorem proving (see [71, Example 3.5] for a broader explanation of importance of gridworlds in RL). Our dataset is intended to become a general purpose testing ground for theorem proving and reinforcement learning methods, in particular for meta-learning and hierarchical learning algorithms.

TPTP [69] consists of 22507 problems in 53 domains collected over several decades. A large dataset for developing machine learning for theorem proving based on the Mizar Mathematical Library (MML) [26] was introduced by [72] in the MPTP project. The dataset was used e.g. by [75, 74, 40, 2]. Similar datasets based on the Isabelle/HOL, HOL Light/Flyspeck and HOL4/CakeML systems and projects [9, 38, 23] were introduced in the last decade and used for the CASC LTB (large theory) ATP competition [70] and other system evaluations. Such datasets cover large areas of mathematics and computer science and contain diverse axioms, lemmas, theorems, definitions, and symbols. Smaller subsets of lemmas leading to the Bolzano-Weierstrass theorem were selected from the MPTP dataset to form the MPTP Challenge [73] and the MPTP2078 benchmark [1]. HOLStep [37] introduced a dataset based on 11400 proofs, including a proof of the Kepler Conjecture [28], formalized using HOL Light [30]. In HOLStep and in FormulaNet [78] the dataset was used as a benchmark for various neural architectures. The recent HOList project [7, 53, 6] uses 29462 theorems formalized in HOL Light and instruments them for experiments oriented towards tactic selection, where a tactic is a human-designed program which aggregates multiple proof steps. GamePad [32] introduced a dataset based on a formalization of the Feit-Thompson Theorem [25] along with a generated algebra problems. It is intended for learning tactic selection together with an auxiliary task of predicting the number of proof steps left. A dataset based on theorems proved in HOL4 [68] was used for developing the TacticToe [24] learning-guided tactical prover. [61] proposed a dataset of simple algebraic problems expressed in English. Arithmetic problems without a natural language context were tackled by Neural GPUs [36] and its improved successors [21]. Supervised learning was also applied to various instances of propositional satisfiability in NeuroSAT [65] and NeuroCore [64] as well as in [17, 12]. Datasets introduced in Section 5 are OpenAI-gym [10] compliant and can be tested with modern RL algorithms. Previous work on theorem proving and RL includes TacticToe, HOList and rlCoP [41]. TacticToe and rlCoP use guided Monte Carlo Tree Search (MCTS) and HOList proposes a custom RL algorithm.

Machine learning systems for guidance of theorem provers. Our current work focuses on providing guidance for the fCoP [42] theorem prover. fCoP is an OCaml implementation of the very compact lean connection tableau prover [52]. fCoP was used as the proof engine in the guided provers FEMaLeCoP [39] and rlCoP [41]. FEMaLeCoP learns only from positive data using two simple, but fast machine learning models (custom nearest neighbour and naive Bayes). In rlCoP, the value and policy functions of the guided MCTS algorithm are learned similar to [4, 66], using

[1]This distribution does not include the fCoP theorem prover, which cannot yet be publicly released, however a binary can be obtained upon request.

gradient boosted trees as implemented in the XGBoost [11] library. In contrast, we use neural network models instead of trees and the Proximal Policy Optimization (PPO) [62] algorithm instead of MCTS. In a longer run we believe that these methods should be combined, see [20, Section 6.2], but in this work we propose to investigate how much can be achieved directly via rollouts and without a search algorithm like MCTS. A distinctive feature of our approach is the ability to perform very long rollouts both in training and evaluation. We demonstrate this in Section 6, see Figure 4. [47, 33, 13, 34] added learning-based guidance to E prover [63]. These are supervised experiments which learn from saturation-style proof traces.

Meta-learning suites of RL environments. Meta-learning algorithms in the context of RL can be tested using a suite of simulated robotic tasks [18, 54], one of discrete environments proposed in [14, 35, 50, 58, 48] or a new MineRL [27] suite of problems with mixed continuous and discrete actions. Our suite of tasks involves discrete actions.

Curriculum learning and reinforcement learning. Our algorithm FLoP is an adaptation of the curriculum learning algorithms presented in [55, 59] in a context of a suite of reinforcement learning environments presented in Section 5.

3 fCoP and the Connection Tableau Calculus

In this section, we give a brief overview of the connection tableau method, as implemented by the fCoP system. We assume basic first-order logic and theorem proving terminology [56]. The input is a (mathematical) problem consisting of *axioms* and *conjectures* formally stated in first-order logic (FOL). The calculus searches for *refutational proofs*, i.e. proofs showing that the axioms together with the negated conjectures are *unsatisfiable*. The FOL formulas are first translated to *clause normal form* (CNF), producing a set of first-order *clauses* consisting of *literals* (atoms or their negations). Figure 1 shows a *closed connection tableau*, i.e., a finished proof tree where every branch contains *complementary literals* (literals with opposite polarity). Since all branches contain a pair of contradictory literals, this shows that the set of clauses is unsatisfiable. Proof search starts with a *start clause* as a *goal* and proceeds by building a connection tableau by repeatedly applying *extension steps* and *reduction steps*.

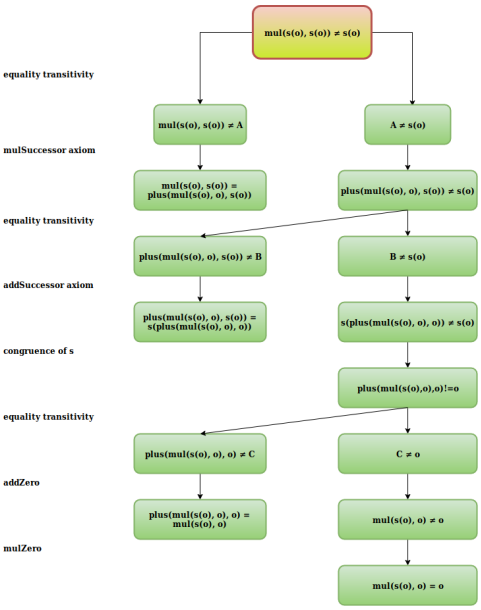


Figure 1: *A closed tableau tree of the proof of* $1 \cdot 1 = 1$. *On the left we list the actions taken in the proof. See* `http://bit.ly/site_atpcurr` *for details.*

The extension step connects (*unifies*) the *current goal* (a selected tip of a tableau branch) with a complementary literal of a new clause. This extends the *current branch*, possibly splitting it into several branches if there are more literals in the new clause, and possibly *instantiating* some variables in the tableau. The reduction step connects the current goal to a complementary literal of the *active path*, thus *closing* the current branch. The proof is finished when all branches are closed. The extension and reduction steps are nondeterministic, requiring backtracking in the standard connection calculus. Brute force search such as *iterative deepening* can be used to ensure completeness, i.e., making sure that the proof search finds a proof if there is any.

fCoP represents theorem proving as a one-person game. The game ends with a success if a proof is found. The prover has many choices to make along the way, hence it typically has to explore a search space that is exponentially large in the length of the proof. In fCoP, the action space is roughly correlated with the size of the axiom set. While this can be large for large problems, typically only a few actions are available in any particular state.

4 FLoP – the Main Algorithm

The FLoP algorithm combines the connection tableau calculus with guidance based on PPO and curriculum learning [55, 59]. Actions in our theorem proving game consist of selecting an extension step as defined in Section 3 (reduction steps are performed automatically by the game engine). Figures 2 and 3 show how actions interact with other components of FLoP. Each extension step involves selecting one of the clauses, however, not all clauses are applicable as actions at a given proof step, due to the unification condition. The full information about the game state consists of all previous proof steps, the partial proof tree (proof state) and the current goal. The state and actions (formulas) are represented using previously developed features [43, 39, 41]. The features include (suitably hashed) triples of adjacent nodes in the formula trees and the partial proof trees, as well as some global features e.g. number of open goals, maximum goal size, maximum goal depth. This means that the proof states and the actions are presented as (sparse) fixed length vectors, see the inputs to the policy and value networks in Figure 2. These features have proved useful but are not free from problems. See the discussion in Section A.

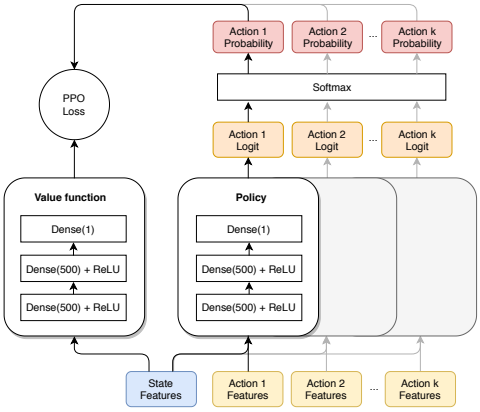


Figure 2: *Value and Policy network architectures in PPO. Their inputs are state and state-action pair features, respectively. The policy returns a score for each action, which are then normalized to a probability.*

Curriculum Learning on Proofs. In theorem proving we are dealing with sparse rewards and we tackle this with the help of curriculum learning as implemented in Algorithm 1.

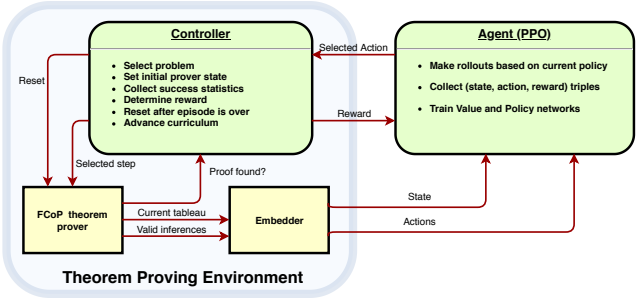


Figure 3: *Theorem proving as a reinforcement learning environment*

First, in line 6 of Algorithm 1 we sample a problem. In lines 7-20 we play an episode: if we have a proof then we start from the state dictated by the global curriculum (lines 7-9). If we do not have a proof then we start from the beginning. If the policy succeeds in finding a proof of a yet unproven problem then we reset the global curriculum to 1 in line 20. We sample k episodes repeating k times the loop in lines 6-20 and finally decide whether to increase the global curriculum in lines 23-24.

We can advance curriculum globally (as in Algorithm 1) or independently for each problem. We found that global advancement makes learning more stable, so that is our default approach. We can start learning with or without training proofs. It does not change the processing of Algorithm 1. In Section 6 we provide experimental evidence with regard to both approaches.

5 Datasets

We introduce a suite of problems in the theory of Robinson Arithmetic [57]. This theory is rather limited, however, it seems to be particularly suited for machine learning methods: solutions are long, repetitive and rather predictable for humans. Still, as we shall see in Section 6, state-of-the-art systems struggle to solve some of these problems. Each problem is a

Table 1: *Axioms of Robinson Arithmetic. These are extended automatically with special axioms for handling equality: reflexivity, symmetry, transitivity as well as three congruence axioms (of s, $plus$ and mul).*

Name	Axiom
zeroSuccessor	$\forall X : \neg(o = s(X))$
diffSuccessor	$\forall X, Y : (s(X) = s(Y)) \Rightarrow (X = Y)$
addZero	$\forall X : plus(X, o) = X$
addSuccessor	$\forall X, Y : plus(X, s(Y)) = s(plus(X, Y))$
mulZero	$\forall X : mul(X, o) = o$
mulSuccessor	$\forall X, Y : mul(X, s(Y)) = plus(mul(X, Y), X)$

Algorithm 1 FLoP: Curriculum Learning on Proofs

Input: problem set $\mathcal{P}$, policy π, progress threshold $\in [0..1]$
train steps $\in \mathbb{N}$, episodes between updates: k $\in \mathbb{N}$
Output: trained policy π, possible new proofs for problems in $\mathcal{P}$

steps $\leftarrow 0$
curriculum $\leftarrow$ *1*
while steps < train steps **do**
  successes $\leftarrow 0$
  for j in 1..k **do**
    $p \leftarrow$ random problem from problem set $\mathcal{P}$ ▷ An episode corresponds to a problem
    if p has stored proof **then** ▷ Determine initial state
      Take proof steps according to stored proof until *curriculum* number of steps remain
      $s_0 \leftarrow$ state of problem p after initial proof steps taken
    else
      $s_0 \leftarrow$ starting state of problem p
    while not episode over **do**
      Take action according to policy $\pi(a_i|s_i)$, observe next state s_{i+1} and reward r_{i+1}
      steps $\leftarrow$ steps + 1
    if proof is found for p **then**
      successes $\leftarrow$ successes + 1
      if found proof is shorter than previous proof **then**
        store proof as new proof for p
      if no proof of p was known before **then**
        curriculum $\leftarrow 1$ ▷ Restart curriculum learning
  Update policy π
  success rate $\leftarrow$ successes / k
  if success rate > progress threshold **then**
    curriculum $\leftarrow$ *curriculum* + 1 ▷ Advance curriculum

separate environment and a guidance system is expected to learn on some of them and then perform well in unseen environments, which is a meta-learning task. The problem sets are intended to be a general purpose testing ground for theorem proving and RL methods, in particular for meta-learning and hierarchical learning algorithms.

Robinson Arithmetic defines basic properties of arithmetic expressions. The signature of the language contains an atom 'o' (representing 0), functions 's', 'plus' and 'mul' (representing $+1$, $+$ and $\cdot$, respectively), and the equality predicate '$=$'. For example, formula $3 \cdot 1 + 2 = 4 + 1$ is written as

$$plus(mul(s(s(s(o))), s(o)), s(s(o))) = plus(s(s(s(s(o)))), s(o)).$$

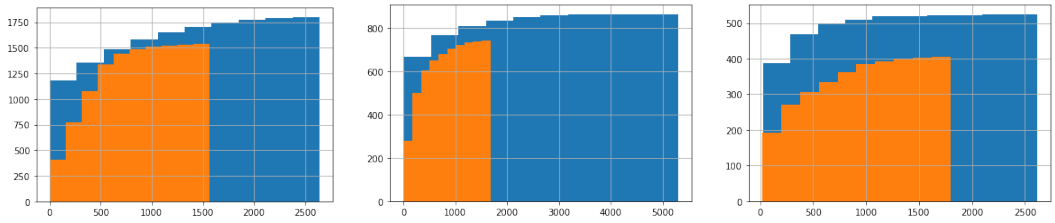
We use the axioms provided in Table 1. The unary representation of numbers (e.g., $s(s(s(o)))$ represents 3) results in large expressions and long proofs as the numbers increase. For example, $((8 + 5) \cdot 8) \cdot 5 = 520$ takes over 16000 steps to prove in fCoP. We show an example of such a proof on the project website. In Table 2 we identify three problem sets of increasing complexity that we use in Section 6 to evaluate FLoP. For Stage 1, a good ordering of the available inference actions is sufficient to find a proof. Stage 2 is harder, as the current goal is also important for selecting an action. For example, the equality reflexivity $A = A$ is usually not needed, except for some cases, due to the unification it triggers. So the system has to learn that this action is useful in particular situations. Stage 3 is much harder, because some of the "rare" actions are tied to global progress in the proof, for example, when we move focus from the left side of the equation to the right side.

6 Experiments

In this section we present six experiments. Experiment 1 demonstrates that FLoP can learn when no proof is provided, only the training problems. In Experiment 2 we compare FLoP with other provers and show that it performs very well on the arithmetic datasets. In Experiment 3 we show that FLoP tends to solve problems in fewer steps than rlCoP and also solves problems that require significantly longer proofs. Experiment 4 shows that FLoP can generalize from a single training problem that

Table 2: *Three challenges defined in the theory of Robinson Arithmetic. The eval set contains all problems for stage* 1 *and is randomly sampled for stages* 2 *and* 3*.*

Stage	Set	Size	Description
Stage 1	Train	2	$1+1=2, 1\cdot 1=1$
	Eval	1800	Expressions of the form $N_1+N_2=N_3$, $N_1\cdot N_2=N_3$, where $0\leq N_i<30$. (Examples: $3{+}4{=}7$ or $5{\cdot}12{=}60$)
Stage 2	Train	3	$1+1=2, 1\cdot 1=1, 1\cdot 1\cdot 1=1$
	Eval	1000	$T=N$, where $0\leq N$, and T is a random expression with 3 operators and operands N_i such that $0\leq N_i<10$. (E.g.: $((3{+}4){\cdot}2){+}6{=}20$)
Stage 3	Train	810	$T_1=T_2$, where T_1 and T_2 are random expressions with 3 operators and operands N_i such that $0\leq N_i<2$.
	Eval	1000	$T_1=T_2$, where T_1 and T_2 are random expressions with 3 operators and operands N_i such that $2\leq N_i<10$. (E.g. $((3{+}4){\cdot}2){+}6{=}((1{+}1){\cdot}5){\cdot}2$)



(a) (b) (c)

Figure 4: *Comparing the length of proofs found by* FLoP *(blue) and* rlCoP *(orange) on the Robinson Arithmetic dataset. All figures are cumulative histograms, vertical axes show the number of proofs, horizontal axes show the length of proofs. Best models are shown for both* FLoP *(based on Experiment 3) and* rlCoP*. Figures (a), (b), (c) correspond to Stage 1, 2, 3 respectively.* FLoP *found more proofs in all stages.*

is provided with a proof. In Experiment 5 we show that proofs of harder problems provide more valuable training signal and we obtain the best generalization when we learn from some longer proofs. Finally Experiment 6 shows that FLoP is more robust than supervised learning on training proofs.

In total we used around 2.5M core-hours on Xeon E5-2680v3 processors, approximately 250-300 core-years. Our hyperparameters were selected using small grid searches. We checked standard RL parameters (e.g., the discount factor), parameters related to curriculum scheduling (e.g., local vs. global), neural network architectures (1–5 layers with 128–1024 neurons), feature sizes (64–1024) and training steps (10^5 – 10^8). Parameters used in the experiments are described in configuration files which are accessible along with the shared codebase.

Each model was trained to achieve 100% accuracy on the training set. During evaluation, the system was allowed to make 100 attempts per problem, each with 60 sec. time limit, without backtracking. We report two evaluation metrics: 1) *Succ.*: percentage of proofs found and 2) *Len.*: average length of proofs found. We have trained 5 models per a set of hyperparameters (unless otherwise noted). Reported numbers are means, with standard deviations in parenthesis. We discuss some failure modes in Appendix A.

Experiment 1: Learning without proofs. We train FLoP with the training problems defined in Section 5, without proofs. The system can find training proofs through exploration and learn from them as we show in Table 3. Curriculum learning performs well for Stage 1 and 2, however in Stage 3 it only solves 3%. This massive overfitting, addressed in Experiments 4 and 5, happens because the system tends to overuse equality congruence axioms, i.e. when trying to prove $a+b=c+d$ or $a\cdot b=c\cdot d$, it often proceeds by reducing the problem to proving

Table 3: *Curriculum learning compared with only exploration based learning.*

	No curriculum		Curriculum	
Stage	Succ.	Len.	Succ.	Len.
1	**1.0** (0.0)	361(2)	**1.0** (0.0)	359(3)
2	0.46(0.09)	88(36)	**0.75** (0.08)	255(93)
3	**0.47** (0.05)	273(27)	0.03(0.03)	289(183)

$a = c$ and $b = d$, which does not work in general. In the training set, all numbers are 0 and 1 and this approach works more often. We also compare curriculum learning with learning based on exploration only. As Figure 5 in the Appendix shows, curriculum learning yields more rewards. This makes no difference in the setting of Stage 1, helps greatly in Stage 2 and results in overfitting in Stage 3.

Experiment 2: Comparison with other Provers. We compare FLoP with a random model – one that assigns equal probability to each valid action –, two state-of-the-art saturation-style theorem provers (E, Vampire), a strong connection tableau prover (leanCoP [52]) one connection tableau prover using learning-based guidance (rlCoP [41]).

Table 4: *Comparing a random model, Vampire, E, leanCoP, rlCoP and FLoP, with respect to success ratio for Stage* 1*,* 2 *and* 3 *problems. Our method (FLoP) is marked in grey. E_1 – auto mode, E_2 – auto-schedule mode, E_3 – auto-schedule with renamed equality.*

Stage	Random	Vampire	E_1	E_2	E_3	leanCoP	rlCoP	FLoP
1	0.04	0.60	0.60	**1.0**	0.54	0.22	0.86	**1.0**
2	0.05	0.40	0.39	**1.0**	0.25	0.14	0.74	0.84
3	0.00	0.34	0.28	**1.0**	0.22	0.01	0.41	0.51

Vampire, E and leanCoP use human-designed strategies instead of learning. In our tests we use the *casc* mode for Vampire, the *auto* and *auto-schedule* modes for E and the default collection of 40 strategies for leanCoP (the standard casc setup), each with a timeout of 60 sec. per problem. For rlCoP we used the same hyperparameters as those described in [41], only modifying the policy temperature from 2.5 to 1.5. The number of inferences in the MCTS was limited to 200000. For Stage 1 and 2 rlCoP was run directly on the evaluation set. For Stage 3 all problems were too hard to solve without guidance within the inference limit, so we started with the version trained on the solutions of Stage 2. For FLoP we report the best models trained without proofs.

The success ratios are given in Table 4. FLoP is only outperformed by E's *auto-schedule*, which tries multiple strategies and finds one with the left-to-right ordering of all the addition and multiplication axioms. This solves all of our problems immediately without any proof search by only using rewriting to a normal form [5]. This demonstrates the power of equational theorem proving when a suitable term ordering exists and can be found by human-designed heuristics. This is, however, far from guaranteed in general and nontrivial even in such simple domains, as witnessed by Vampire's failure to find this ordering. To evaluate E without access to its built-in rewriting capability, we have renamed the equality to a new predicate 'eq' axiomatized exactly in the same way as in fCoP. The auto-schedule mode then becomes somewhat weaker than the auto mode, see Table 4.

Experiment 3: FLoP vs. rlCoP with Respect to Proof Lengths. Due to the same underlying calculus, the proofs found by rlCoP and FLoP are directly comparable and it is insightful to compare them with respect to the length of proofs. Figure 4 shows that FLoP manages to solve more problems, and even finds some very long proofs. This is, however, not because FLoP's proofs are unnecessarily long: we demonstrate in Table 5 that FLoP tends to find shorter proofs for problems solved by both systems. It is interesting to note that out of the 351 problems solved by both, none had the same length, which suggests that the provers acquired different strategies.

Table 5: *Comparing rlCoP and FLoP with respect to proofs found and proof lengths on Stage* 3*.*

	found	found only	found by both	shorter proof
rlCoP	406	55	351	53
FLoP	**517**	**166**	351	**298**

Experiment 4: Learning from a Single Problem with Proof Provided. When the proof of training problems is available, FLoP can use it to make learning more efficient. In this experiment, FLoP is restricted to learn from a single, rather simple training problem, but we also provide its proof. Table 6 shows that FLoP generalizes very well in this setup, expecially in Stage 1 and 2.

Table 6: *Curriculum Learning on a single training proof. Numbers with $^{\#}$ and * are based on* 1*,* 2 *runs.*

Stage	Training problem	Succ.	Len.
1	$1 \cdot 1 = 1$	1.0(0)	368(5)
2	$1 \cdot 1 \cdot 1 = 1$	$0.71(0.01)^{\star}$	$311(61)^{\star}$
3	$1 \cdot 1 \cdot 1 + 1 = 1 \cdot 1 + 0 + 1$	$0.37^{\#}$	$336^{\#}$

Table 7: *Curriculum learning for Stage 3 on two harder problems with proofs of* 113 *and* 108 *steps. Results are based on* 3 *runs.*

Training problem	Succ.	Len.
$1 \cdot 2 + 1 + 1 = (1 + 1) \cdot 1 \cdot 2$	0.32(0.05)	566(14)
$1 \cdot 2 + 1 + 1 = (1 + 1) \cdot 1 \cdot 2$ $(1 + 1 + 1) \cdot 2 = 2 \cdot 1 + 2 + 2$	**0.51** (0.03)	590(54)

Experiment 5: Learning from long proofs. The massive overfitting in Stage 3 shown in Figure 5 is endemic to modern RL methods [79]. Here, in particular, the system tends to overuse equality congruence axioms, i.e. when trying to prove $a + b = c + d$ or $a \cdot b = c \cdot d$, it often proceeds by reducing the problem to proving $a = c$ and $b = d$, which does not work in general. In

the training set, all numbers are 0 and 1 and this approach works more often. The harder the problems, the less likely they can be solved with such heuristic approaches, hence harder training problems promise more valuable training signal. We demonstrate this by training FLoP on a few selected harder problems with proofs provided. A single longer training proof (113 steps) is sufficient to discourage the overuse of equality axioms. Adding one more training problem (108 steps) helps even more and we obtain the best model for Stage 3, see Table 7. Also note the huge increase in the length of proofs: this is partly because we solve new hard problems and partly because the system resorts to longer but safer proof strategies.

Table 8: *Curriculum Learning vs Supervised Learning on Stage* 1 *and* 2*, using training proofs with some* 1 - 3 *steps added for distraction. FLoP is barely affected, while supervised learning's performance degrades. Numbers with* * *are averaged from 2 runs.*

Stage	Proof Lengths	Supervised		Curriculum	
		Succ.	Len.	Succ.	Len.
1	5, 9	0.98(0.04)	327(58)	**1 (0.01)**	363(5)
1	7, 10	**1 (0)**	359 (0)	0.98(0.01)	327(18)
1	9, 11	0.52(0.08)	54(11)	**0.98 (0.01)**	340(18)
2	5, 9, 23	**0.85 (0.04)**	377(47)	0.76(0.02)*	291(16)*
2	7, 10, 24	**0.74 (0.04)**	433(110)	0.71(0.01)*	311(61)*
2	9, 11, 25	0.59(0.08)	193(49)	**0.76 (0.01)***	267(109)*

Experiment 6: Curriculum Learning vs Supervised Learning. When training proofs are available, a natural baseline of curriculum learning is supervised learning on the proof steps. While such behavioral cloning sometimes leads to great performance, we show in Table 8 that it greatly depends on the quality of the given proof. For the three problems in the training set of Stage 1 and 2, we take the shortest proofs (5, 9 and 23 steps) and construct variants with 1-3 extra steps added. We observe that supervised learning degrades as superfluous steps are introduced, while FLoP's exploration allows the system to recover and find the original proofs.

7 Conclusion and Future Work

We have built FLoP, a proof guidance system based on reinforcement learning addressing the problem of finding long proofs in an exponential search space. Previous work [77, 44] focused on finding long proofs with the help of human-designed heuristics. We find that curriculum learning is a suitable approach as it strikes a good balance between exploration and memorization on a suite of arithmetic RL problems introduced in Section 5, allowing our system to generalize from small training problems to larger ones with similar structure. We have created a suite of RL environments based on Robinson arithmetic that contains problems of highly related structure.

Overfitting in Reinforcement Learning is a a well known and largely unsolved problem and we believe that our work offers an interesting new angle on the problem. We show that the greater reward efficiency provided by curriculum learning can result in catastrophic overfitting. Figure 5 and Table 3 show that allowing or disallowing the curriculum can be considered as a trade-off between more efficient training and a higher risk of overfitting. Furthermore, we also show in Table 7 that when we have some training proofs of harder problems, we can greatly reduce overfitting.

This work uses a human-designed representation of formulas similar to one used earlier in [76, 39, 41] and a straightforward encoding of actions. We believe that learned embeddings as well as more sophisticated RL ideas employed before in the context of text games [15, 29, 45, 22, 31, 49] will positively impact the performance of FLoP. We also see potential in exploring other ways of curriculum learning: while in this paper curriculum is on the number of steps to the end of the proof, one could order training problems from easier to more complex ones.

We believe that the learning loop implemented in Algorithm 1 can benefit from integration with memory-based meta-learning methods [60, 51]. One can also look for inspiration from robotics [19] with regard to automatic discovery of curricula of easier problems. In mathematical practice it is quite often the case that instead of proving a conjecture, a similar, but simpler problem is solved, which eventually – maybe over a longer span of time – contributes to the solution of the main conjecture. This encourages us to combine FLoP with Hindsight Experience Replay [3], which utilizes a similar strategy of allowing easier goals in order to ultimately solve a harder goal in the domain of robotics.

Finally we find it interesting to instrument the Bolzano-Weierstrass theorem and its 252 auxiliary lemmas as an RL challenge where the system would be supposed to derive the theorem and all lemmas from scratch using in each derivation only basic axioms, hence forcing long proofs. This can be considered as an RL follow-up to the earlier MPTP Challenge [73].

8 Acknowledgments

Adrián Csiszárik and Zsolt Zombori were supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002), as well as by the Hungarian National Excellence Grant 2018-1.2.1-NKP-00008. The work of Henryk Michalewski was supported by the Polish National Science Center grant UMO-2018/29/B/ST6/02959. Cezary Kaliszyk was supported by ERC grant no. 714034 *SMART*. Josef Urban was supported by the *AI4REASON* ERC Consolidator grant number 649043, and by the Czech project AI&Reasoning CZ.02.1.01/0.0/0.0/15_003/0000466 and the European Regional Development Fund.

This research was supported by the PL-Grid Infrastructure. In particular, quantitative results of FLoP reported in this paper were performed using the Prometheus supercomputer, located in the Academic Computer Center Cyfronet in the AGH University of Science and Technology in Kraków, Poland.

References

[1] Jesse Alama, Tom Heskes, Daniel Kühlwein, Evgeni Tsivtsivadze, and Josef Urban. Premise selection for mathematics by corpus analysis and kernel methods. *J. Autom. Reasoning*, 52(2):191–213, 2014.

[2] Alexander A. Alemi, François Chollet, Niklas Een, Geoffrey Irving, Christian Szegedy, and Josef Urban. Deepmath - Deep Sequence Models for Premise Selection. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2243–2251, USA, 2016. Curran Associates Inc.

[3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *CoRR*, abs/1707.01495, 2017.

[4] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *CoRR*, abs/1705.08439, 2017.

[5] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

[6] Kshitij Bansal, Sarah M. Loos, Markus N. Rabe, and Christian Szegedy. Learning to reason in large theories without imitation. *CoRR*, abs/1905.10501, 2019.

[7] Kshitij Bansal, Sarah M. Loos, Markus N. Rabe, Christian Szegedy, and Stewart Wilcox. HOList: An environment for machine learning of higher-order theorem proving (extended version). *CoRR*, abs/1904.03241, 2019.

[8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 41–48. ACM, 2009.

[9] Jasmin Christian Blanchette, David Greenaway, Cezary Kaliszyk, Daniel Kühlwein, and Josef Urban. A learning-based fact selector for Isabelle/HOL. *J. Autom. Reasoning*, 57(3):219–244, 2016.

[10] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym. *CoRR*, abs/1606.01540, 2016.

[11] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[12] Karel Chvalovsky. Top-down neural model for formulae. In *International Conference on Learning Representations*, 2019.

[13] Karel Chvalovský, Jan Jakubuv, Martin Suda, and Josef Urban. ENIGMA-NG: efficient neural and gradient-boosted inference guidance for E. *CoRR*, abs/1903.03182, 2019.

[14] Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *CoRR*, abs/1812.02341, 2018.

[15] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew J. Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. TextWorld: A learning environment for text-based games. *CoRR*, abs/1806.11532, 2018.

[16] Jeffrey L. Elman. Learning and development in neural networks: the importance of starting small. *Cognition*, 48:71–99, 1993.

[17] Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. Can neural networks understand logical entailment? In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[18] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017.

[19] Carlos Florensa, David Held, Markus Wulfmeier, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. *CoRR*, abs/1707.05300, 2017.

[20] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *CoRR*, abs/1811.12560, 2018.

[21] Karlis Freivalds and Renars Liepins. Improving the neural GPU architecture for algorithm learning. *CoRR*, abs/1702.08727, 2017.

[22] Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. What can you do with a rock? affordance extraction via word embeddings. *CoRR*, abs/1703.03429, 2017.

[23] Thibault Gauthier and Cezary Kaliszyk. Premise selection and external provers for HOL4. In Xavier Leroy and Alwen Tiu, editors, *Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15-17, 2015*, pages 49–57. ACM, 2015.

[24] Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar, and Michael Norrish. Learning to prove with tactics. *CoRR*, abs/1804.00596, 2018.

[25] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A machine-checked proof of the odd order theorem. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, volume 7998 of *LNCS*, pages 163–179. Springer, 2013.

[26] Adam Grabowski, Artur Kornilowicz, and Adam Naumowicz. Mizar in a nutshell. *J. Formalized Reasoning*, 3(2):153–245, 2010.

[27] William H. Guss, Cayden Codel, Katja Hofmann, Brandon Houghton, Noburu Kuno, Stephanie Milani, Sharada Prasanna Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, Manuela Veloso, and Phillip Wang. The MineRL competition on sample efficient reinforcement learning using human priors. *CoRR*, abs/1904.10079, 2019.

[28] Thomas Hales, Mark Adams, Gertrud Bauer, Dat Tat Dang, John Harrison, Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Tat Nguyen, Truong Quang Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Ta, Tran Trung, Diep Thi Trieu, and Roland Zumkeller. A formal proof of the Kepler conjecture. *Forum of Mathematics, Pi*, 5, 01 2015.

[29] Matan Haroush, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. Learning how not to act in text-based games. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018.

[30] John Harrison. HOL Light: A tutorial introduction. In Mandayam K. Srivas and Albert John Camilleri, editors, *Formal Methods in Computer-Aided Design, First International Conference, FMCAD '96, Palo Alto, California, USA, November 6-8, 1996, Proceedings*, volume 1166 of *LNCS*, pages 265–269. Springer, 1996.

[31] Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. Deep reinforcement learning with an unbounded action space. *CoRR*, abs/1511.04636, 2015.

[32] Daniel Huang, Prafulla Dhariwal, Dawn Song, and Ilya Sutskever. GamePad: A Learning Environment for Theorem Proving. In *International Conference on Learning Representations*, 2019.

[33] Jan Jakubuv and Josef Urban. ENIGMA: efficient learning-based inference guiding machine. In Herman Geuvers, Matthew England, Osman Hasan, Florian Rabe, and Olaf Teschke, editors, *Intelligent Computer Mathematics - 10th International Conference, CICM 2017, Edinburgh, UK, July 17-21, 2017, Proceedings*, volume 10383 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 2017.

[34] Jan Jakubuv and Josef Urban. Hammering Mizar by learning clause guidance. *CoRR*, abs/1904.01677, 2019.

[35] Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle tower: A generalization challenge in vision, control, and planning. *CoRR*, abs/1902.01378, 2019.

[36] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[37] Cezary Kaliszyk, François Chollet, and Christian Szegedy. HolStep: A machine learning dataset for higher-order logic theorem proving. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[38] Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *J. Autom. Reasoning*, 53(2):173–213, 2014.

[39] Cezary Kaliszyk and Josef Urban. FEMaLeCoP: Fairly efficient machine learning connection prover. In Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, 2015, Proceedings*, volume 9450 of *LNCS*, pages 88–96. Springer, 2015.

[40] Cezary Kaliszyk and Josef Urban. MizAR 40 for Mizar 40. *J. Autom. Reasoning*, 55(3):245–256, 2015.

[41] Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olsák. Reinforcement learning of theorem proving. In *NeurIPS*, pages 8836–8847, 2018.

[42] Cezary Kaliszyk, Josef Urban, and Jiři Vyskočil. Certified connection tableaux proofs for HOL Light and TPTP. In *Proceedings of the 2015 Conference on Certified Programs and Proofs*, CPP '15, pages 59–66. ACM, 2015.

[43] Cezary Kaliszyk, Josef Urban, and Jiří Vyskočil. Efficient semantic features for automated reasoning over large theories. In Qiang Yang and Michael Wooldridge, editors, *Proc. of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, pages 3084–3090. AAAI Press, 2015.

[44] Michael K. Kinyon, Robert Veroff, and Petr Vojtechovský. Loops with abelian inner mapping groups: An application of automated deduction. In Maria Paola Bonacina and Mark E. Stickel, editors, *Automated Reasoning and Mathematics - Essays in Memory of William W. McCune*, volume 7788 of *LNCS*, pages 151–164. Springer, 2013.

[45] Bartosz Kostka, Jaroslaw Kwiecien, Jakub Kowalski, and Pawel Rychlikowski. Text-based adventures of the golovin AI agent. *CoRR*, abs/1705.05637, 2017.

[46] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In *CAV*, 2013.

[47] Sarah M. Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep network guided proof search. In *21st International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, 2017.

[48] Jacob Menashe and Peter Stone. Escape room: A configurable testbed for hierarchical reinforcement learning. *CoRR*, abs/1812.09521, 2018.

[49] Karthik Narasimhan, Tejas D. Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. *CoRR*, abs/1506.08941, 2015.

[50] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in RL. *CoRR*, abs/1804.03720, 2018.

[51] Pedro A. Ortega, Jane X. Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alexander Pritzel, Pablo Sprechmann, Siddhant M. Jayakumar, Tom McGrath, Kevin Miller, Mohammad Gheshlaghi Azar, Ian Osband, Neil C. Rabinowitz, András György, Silvia Chiappa, Simon Osindero, Yee Whye Teh, Hado van Hasselt, Nando de Freitas, Matthew Botvinick, and Shane Legg. Meta-learning of sequential strategies. *CoRR*, abs/1905.03030, 2019.

[52] Jens Otten and Wolfgang Bibel. leanCoP: lean connection-based theorem proving. *J. Symb. Comput.*, 36:139–161, 2003.

[53] Aditya Paliwal, Sarah M. Loos, Markus N. Rabe, Kshitij Bansal, and Christian Szegedy. Graph representations for higher-order logic and theorem proving. *CoRR*, abs/1905.10006, 2019.

[54] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *CoRR*, abs/1903.08254, 2019.

[55] Cinjon Resnick, Roberta Raileanu, Sanyam Kapoor, Alex Peysakhovich, Kyunghyun Cho, and Joan Bruna. Backplay: "Man muss immer umkehren". *CoRR*, abs/1807.06919, 2018.

[56] Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning*. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 2001.

[57] Raphael M. Robinson. An essentially undecidable axiom system. *Proceedings of the International Congress of Mathematics*, pages 729–730, 1950.

[58] Melrose Roderick, Christopher Grimm, and Stefanie Tellex. Deep abstract q-networks. *CoRR*, abs/1710.00459, 2017.

[59] Tim Salimans and Richard Chen. Learning Montezuma's Revenge from a single demonstration. *CoRR*, abs/1812.03381, 2018.

[60] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. Meta-learning with memory-augmented neural networks. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1842–1850. JMLR.org, 2016.

[61] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *CoRR*, abs/1904.01557, 2019.

[62] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[63] Stephan Schulz. System Description: E 1.8. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Proc. of the 19th LPAR*, volume 8312 of *LNCS*. Springer, 2013.

[64] Daniel Selsam and Nikolaj Bjørner. Neurocore: Guiding high-performance SAT solvers with unsat-core predictions. *CoRR*, abs/1903.04671, 2019.

[65] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. *CoRR*, abs/1802.03685, 2018.

[66] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017.

[67] B. F. Skinner. Teaching machines. *Science*, 128(3330):969–977, 1958.

[68] Konrad Slind and Michael Norrish. A brief overview of HOL4. In Otmane Aït Mohamed, César A. Muñoz, and Sofiène Tahar, editors, *Theorem Proving in Higher Order Logics, 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceedings*, volume 5170 of *LNCS*, pages 28–32. Springer, 2008.

[69] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.

[70] Geoff Sutcliffe and Josef Urban. The CADE-25 automated theorem proving system competition - CASC-25. *AI Commun.*, 29(3):423–433, 2016.

[71] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[72] Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *J. Autom. Reasoning*, 37(1-2):21–43, 2006.

[73] Josef Urban. The MPTP Challenge. `http://www.tptp.org/Seminars/MizarVerification/TheMPTPChallenge.html`, 2006. Accessed: 2019-05-20.

[74] Josef Urban. MaLARea: a Metasystem for Automated Reasoning in Large Theories. In Geoff Sutcliffe, Josef Urban, and Stephan Schulz, editors, *Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories, Bremen, Germany, 17th July 2007*, volume 257 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[75] Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jirí Vyskocil. MaLARea SG1- machine learner for automated reasoning with semantic guidance. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *LNCS*, pages 441–456. Springer, 2008.

[76] Josef Urban, Jirí Vyskocil, and Petr Stepánek. MaLeCoP: Machine learning connection prover. In Kai Brünnler and George Metcalfe, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - 20th International Conference, TABLEAUX 2011, Bern, Switzerland, July 4-8, 2011. Proceedings*, volume 6793 of *LNCS*, pages 263–277. Springer, 2011.

[77] Robert Veroff. Using hints to increase the effectiveness of an automated reasoning program: Case studies. *J. Autom. Reasoning*, 16(3):223–239, 1996.

[78] Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2783–2793, 2017.

[79] Chiyuan Zhang, Oriol Vinyals, Rémi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *CoRR*, abs/1804.06893, 2018.

Appendix A Failure Modes

Despite the apparent simplicity of our arithmetic learning environments, a learning system aiming to solve them has to overcome some hard challenges. We have decided to describe these challenges in detail as they are present in other domains as well, even if it may be harder to detect.

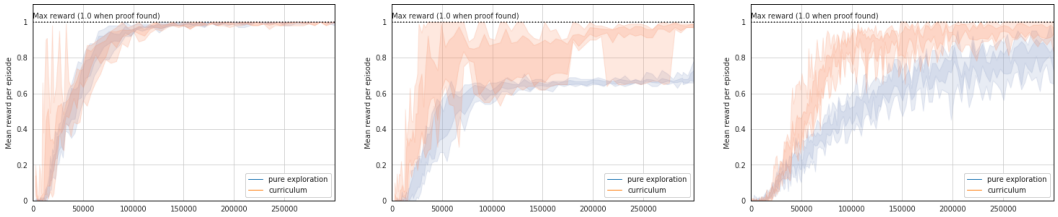
Failure type 1. The reward mechanism of our RL system is biased towards shorter proofs. However, many problems have "shortcuts" that allow for shorter proofs, but that do not generalize well. Consider formula $(1 + 1) + (2 \cdot 2) = (0 + 2) + 4$. There are two ways to prove this equality: 1) compute the values of the expressions on both sides of the equation and notice that they are the same or 2) show that $1 + 1 = 0 + 2$ and $2 \cdot 2 = 4$. The former generalizes better, but the latter results in a shorter proof. Hence, training on this problem might negatively affect the performance of the prover. This is what causes the failure in Experiment 3: through manual inspections of discovered proofs we have concluded that curriculum learning is more efficient at finding and learning shorter proofs of the training problems and it overfits to them.

Failure mode 2. fCoP features do not take into account the order of the arguments of a function, hence $f(a, b)$ and $f(b, a)$ have the same features. This is particularly problematic for Stage 3, since $A = B$ and $B = A$ require different inferences. We addressed this problem by 1) extending state features with those of the preceding action as a substitute of a memory, 2) modified the features to include argument order.

Failure mode 3. Some "rare" events are hard to generalize, because the system sees very few relevant samples during training. This is the case with applying commutativity of equality (replacing $A = B$ with $B = A$), which is only required in Stage 3 and ideally only once per proof, when we move focus from one side of the equation to the other. In Experiment 4, when we trained on a single longer proof, we have noticed that the system was very unsure about this action which resulted in many failed proof attempts. Adding another training proof as enough to overcome this and success score increased from 32% to 51%.

Appendix B The Effect of Curriculum Learning

Figure 5 shows that curriculum learning yields more rewards. This makes no difference in the simple setting of Stage 1, helps greatly in Stage 2 and results in fatal overfitting in Stage 3.



(a) (b) (c)

Figure 5: *(a)-(c) – Stages 1-3, training graphs centered at the mean reward, darker bars are delimited by quantiles at 0.25 and 0.75, lighter bars extending from min to max; in total 36 models, 6 models per graph, 20M samples per experiment. Curriculum helps in Stages 2 and 3.*