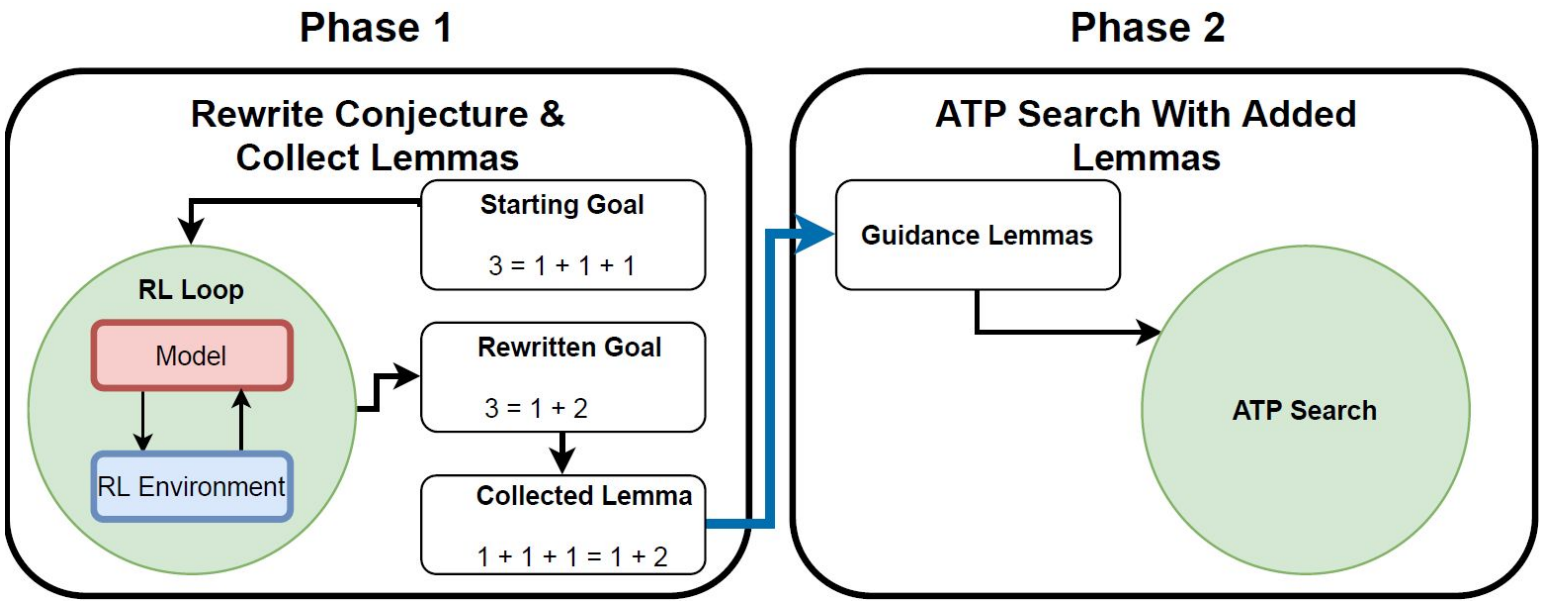
Guiding An Automated Theorem Prover with Neural Rewriting

Jelle Piepenbrock, Tom Heskes, Mikoláš Janota & Josef Urban

Automated Theorem Proving

- The given clause loop algorithm can produce many irrelevant clauses and the the search can get derailed
- Can we guide a theorem prover to improve performance?
- The specific way we chose to try to improve performance is by using a neural network guided rewriter to come up with lemmas that bring the two sides of an equality “closer”.

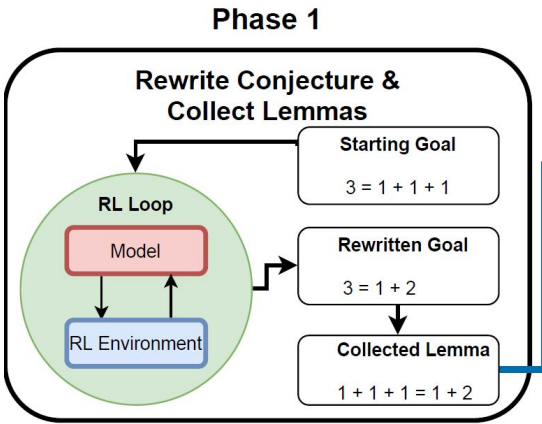
Overall Setup



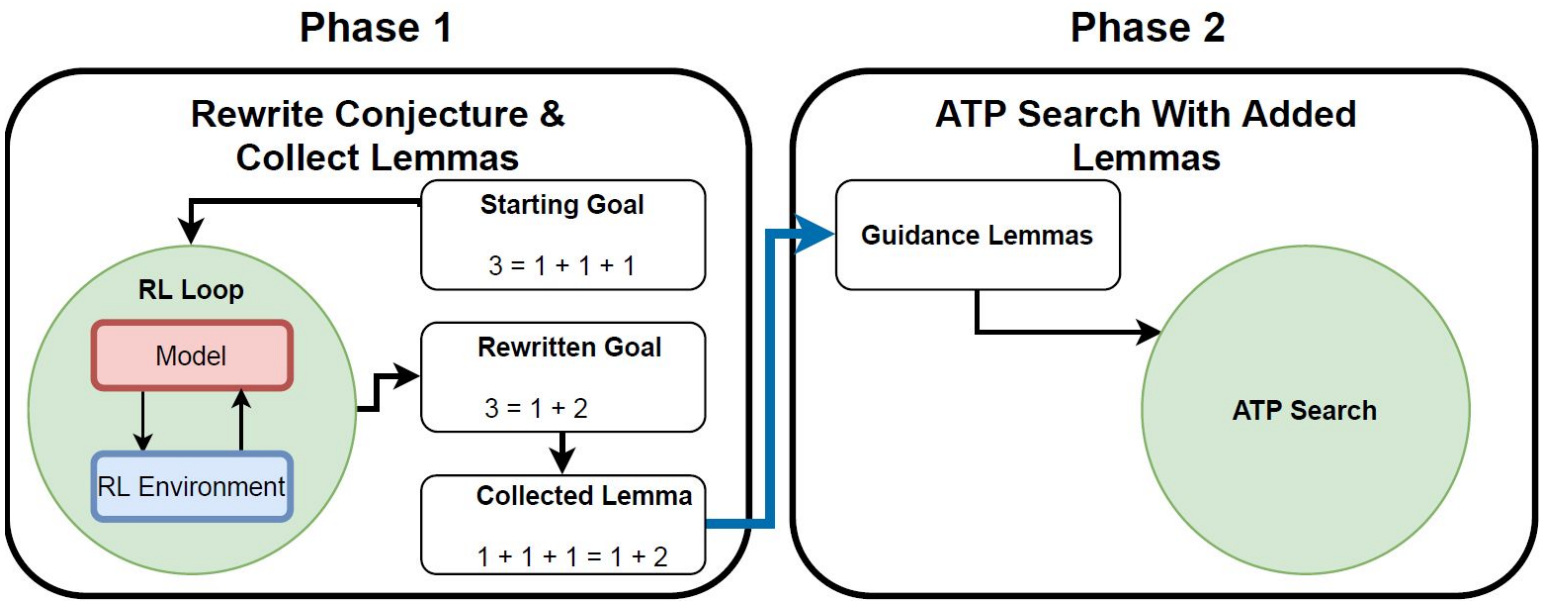
What setting are we in for Phase 1?

- We are using a dataset constructed for (Brown 2020), which has an associated simple prover / RL environment called AIMLEAP.
- It contains ~3500 theorems that were extracted from hint-guided proof attempts for the AIM conjecture in the Prover9 ATP.
- In phase 1, we are purely in an equational setting, where we have a goal statement consisting of LHS and RHS, and we are rewriting both sides until they are unifiable.

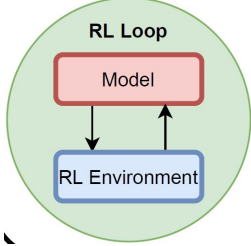
$$T(T(T(x,T(x,y)\backslash 1),T(x,y)\backslash 1),y) = T((T(x,y)\backslash 1)\backslash 1,T(x,y)\backslash 1)$$



Overall Setup

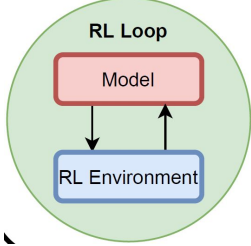


RL formulation for AIMLEAP



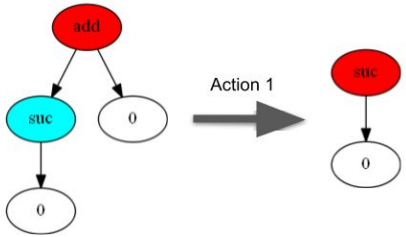
- Tree data structure
- The neural network can move a cursor down the tree from the root and choose one of 87 rewrite rules to apply.
- We have an RL task with 3 (cursor movements)
- 3 + 2 * 87 = 177 possible actions.
- Only reward when there is a proof as decided by AIMLEAP.
- But it’s very hard to stumble upon a proof with 177 possible actions

$$T(T(T(x,T(x,y)\backslash 1),T(x,y)\backslash 1),y) = T((T(x,y)\backslash 1)\backslash 1,T(x,y)\backslash 1)$$



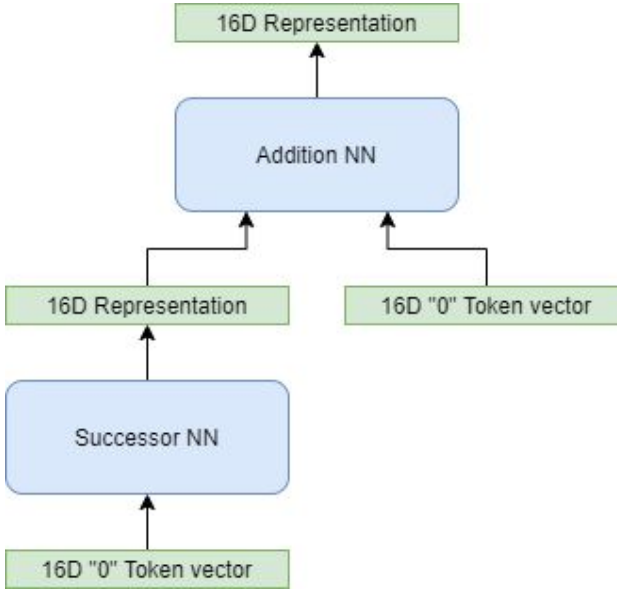
A simpler task

- Robinson arithmetic rewriting
- Dataset of ~5000 problems, from (Gauthier 2018).
- Task is to rewrite a Robinson arithmetic tree to a form where there are only successor nodes (i.e. calculate the value of the expression).
- There are 7 rewrite actions and 2 cursor moves (9 total).
- We have a curriculum of problems, with heuristic for difficulty

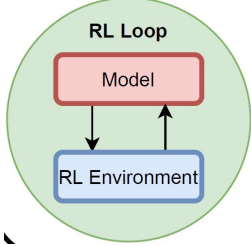


Neural Network Architecture

- Tree neural network
- Each operation is its own multi-layer perceptron subnetwork.
- For RA, we have the successor function, *, +, and one 0 constantr
- For the loop theory, we have L, R, T, K, a, /, \, *, and e.
- After creating an embedding, a 3-layer network decides the action and (if applicable) the value for RL purposes.
- Cursor is also a node

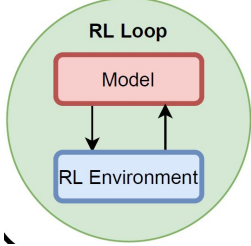


How to train the network?

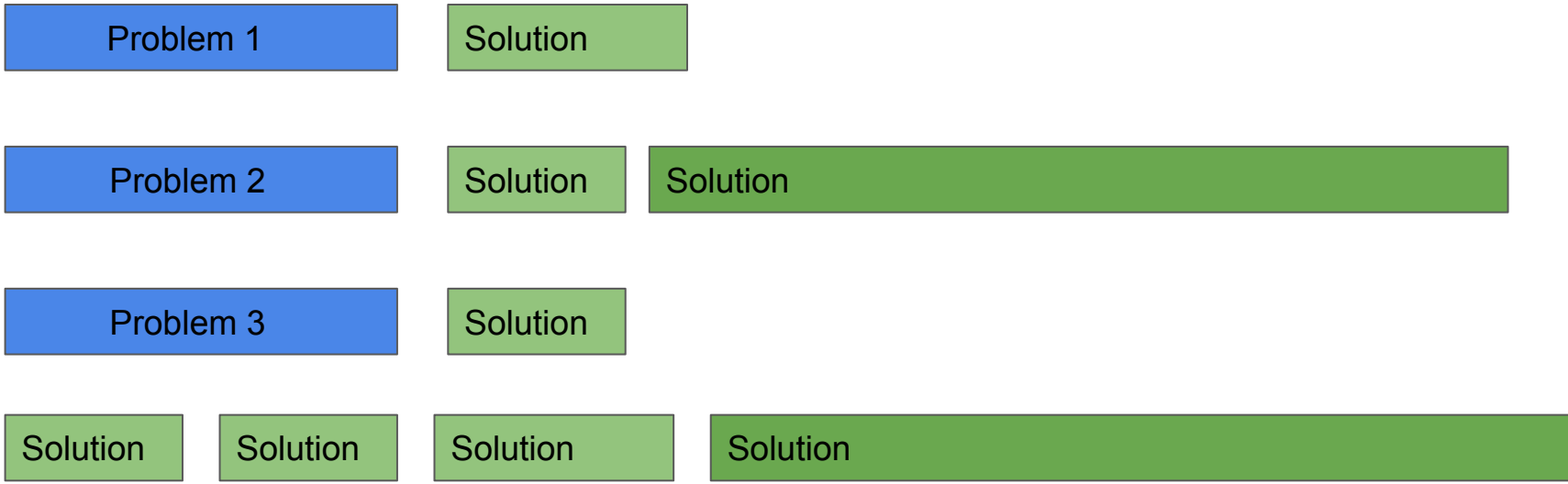


- We still found that it was hard for several RL algorithms to effectively learn this relatively simple Robinson arithmetic task.
- Several baseline RL algorithms, such as PPO and ACER, did not learn effectively.
- Inspecting the episodes and data generated, we found that even if a solution was found for a problem, it was often “forgotten”. We also the solution could vary a lot in length (per problem and between).

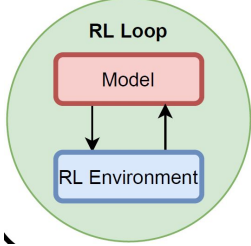
Training method



- We came up with an approach that we call **3SIL**, for stratified shortest solution imitation learning.
- Solutions are episodes that solved the RL task, i.e. lists of state-action pairs (transitions)



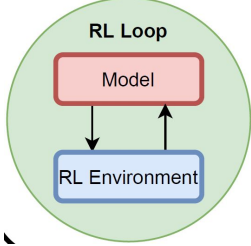
Training method



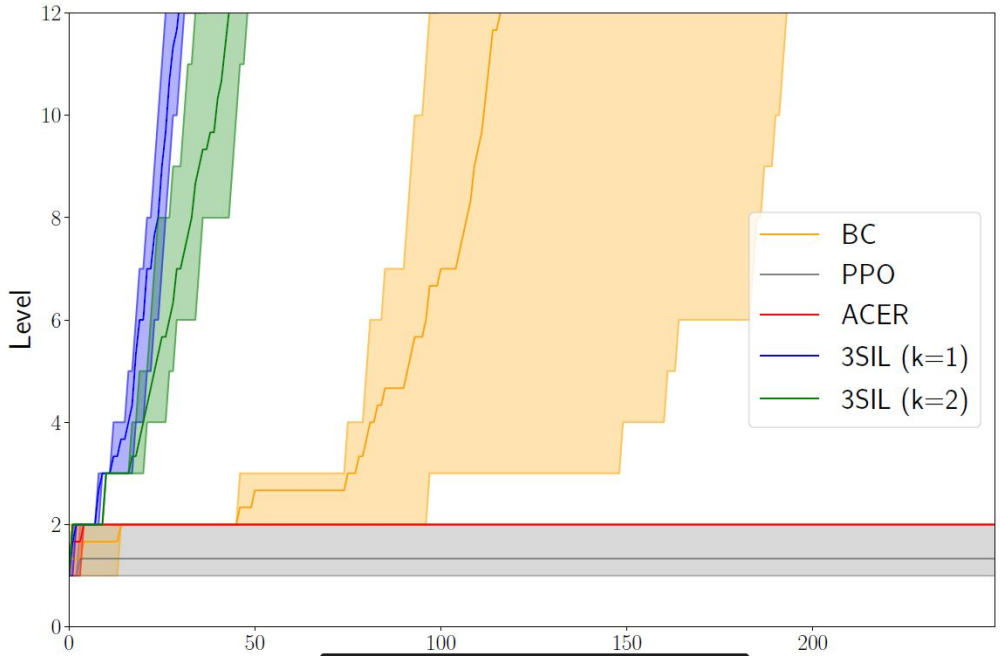
- We came up with an approach that we call 3SIL, for stratified shortest solution imitation learning.
- Solutions are episodes that solved the RL task, i.e. lists of state-action pairs (transitions)

Problem 1	Solution
Problem 2	Solution
Problem 3	Solution

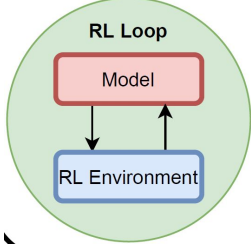
Results for RA



- Curriculum: 400 problems per ‘level’
- 1000 episodes training per epoch
- Test after each epoch, if > 0.95 solved, go to next level. 400 extra problems can now be encountered.

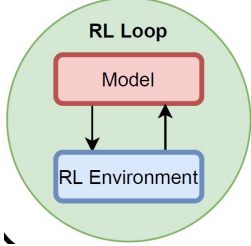


Comments

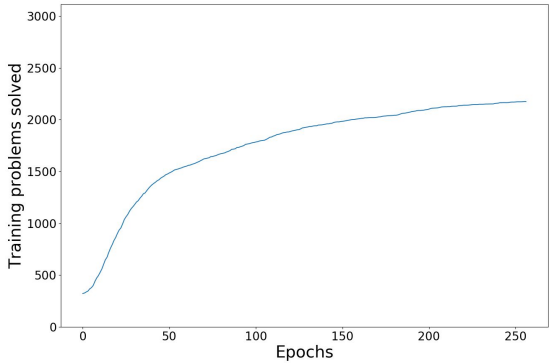


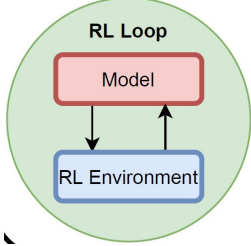
- From this, we concluded that this approach might be good enough to learn the AIMLEAP environment

AIMLEAP Setup



- The randomly initiated policy almost never finds a proof, so we can end up with a tiny amount of proofs to learn from, and starting with too few proofs makes the model very limited
- Before training, we collected 2 million episodes, which results in about 300 theorems proved. We seeded the process with these. This proved sufficient to find solutions for ~2200 out of 3100 training problems.



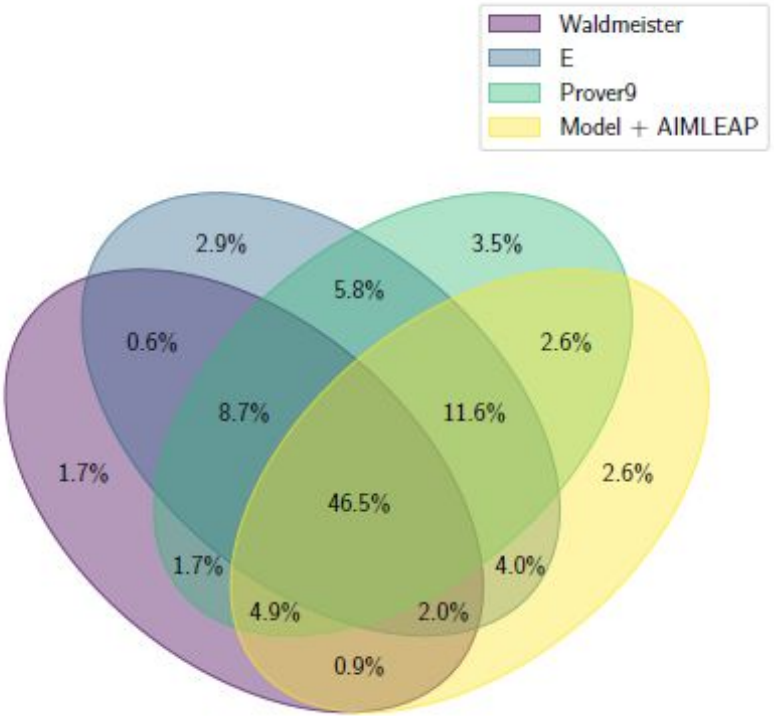


Comparison with ATPs

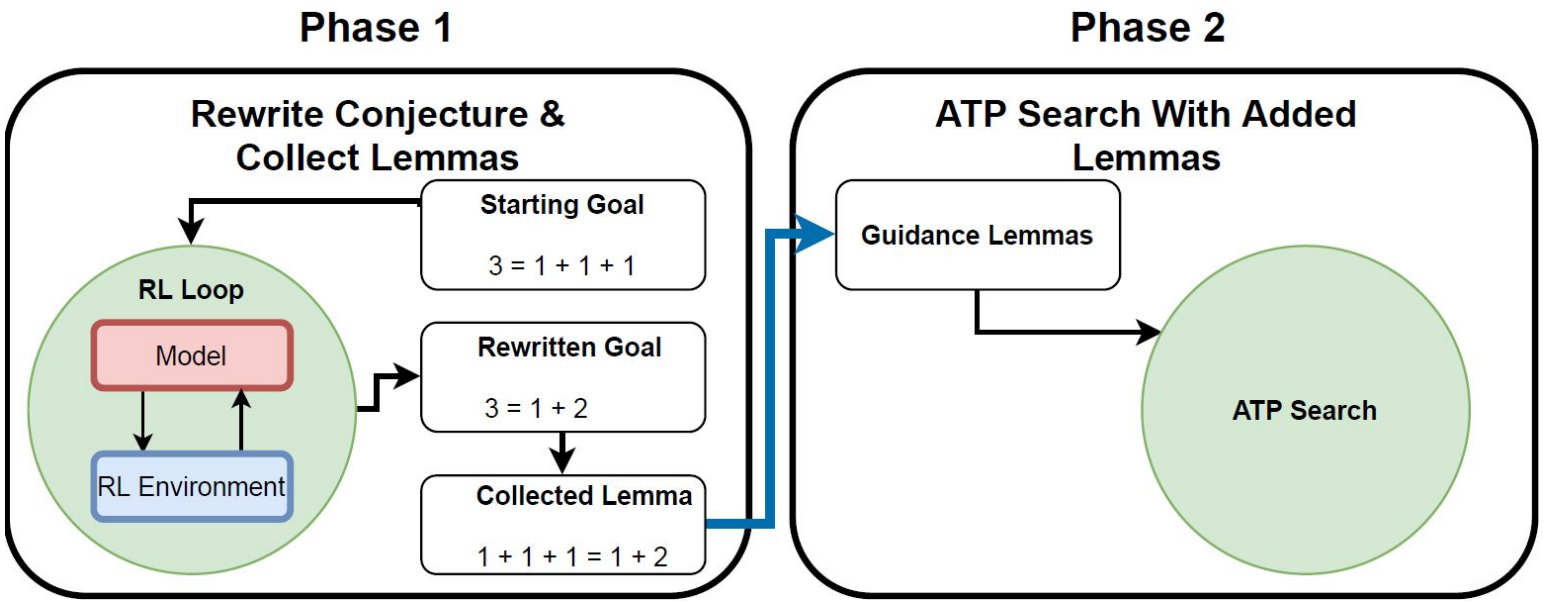
- Results on a held-out test set of the AIMLEAP problems
- Prover9 is the best prover.
- 1 evaluation of the model, which takes under 1 second, solves 58%
- Model with noisy evaluation, for 60s, restarted every 30 steps, finds 70.2% and beats Waldmeister.
- Model does no backtracking

METHOD	SUCCESS RATE
PROVER9 (60S)	0.833
E (60S)	0.802
MODEL (60S)	0.702 ± 0.015
WALDMEISTER (60S)	0.655
MODEL (1X)	0.586 ± 0.029

Proof overlap

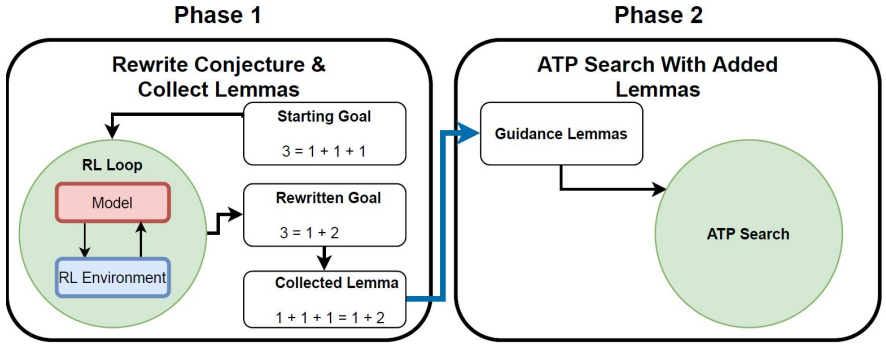


End of phase 1

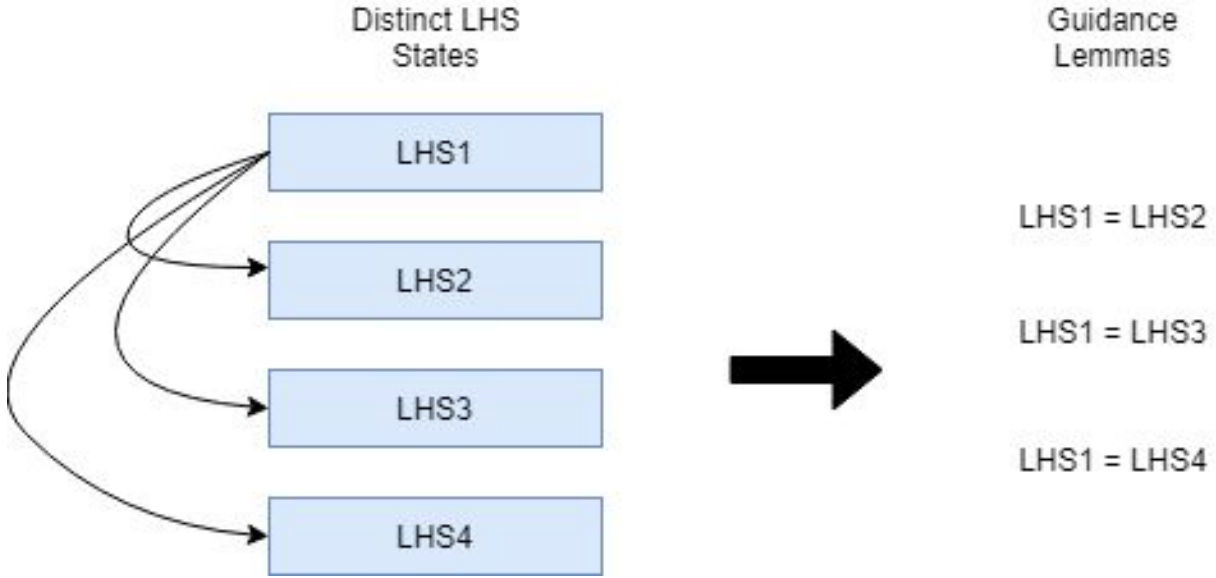


Extracting lemmas

- The LHS and RHS of the conjecture get rewritten in the AIMLEAP environment according to the rewrite rules
- This means we have a set of equalities for both sides (“shortcuts”)
- We add these lemmas as the input for the ATP Prover9



Adding Lemmas



Results

- Results on test set

METHOD	SUCCESS RATE
PROVER9 (1S)	0.715
PROVER9 (2S)	0.746
PROVER9 (60S)	0.833
NEURAL REWRITING (1S) + PROVER9 (1S)	0.841 ± 0.019
NEURAL REWRITING (1S) + PROVER9 (59S)	$\mathbf{0.902} \pm 0.016$

Conclusions

- It is possible to guide ATPs in the equational setting by using a neural network model to suggest useful rephrasings of the conjecture
  - About 7% performance increase of Prover9
- We have shown that the 3SIL (stratified shortest solution imitation) approach can be used to train a neural network within the AIMLEAP environment that is strong enough to compete with provers such as Waldmeister on a specific task.

Future Work

- Different rewriting tasks (suggestions?)
- Metalearning: can we do better than just the shortest proof?
    - Are there proofs with “more generalizable” steps?