# The Isabelle ENIGMA

**Zarathustra A. Goertzel**
Czech Technical University in Prague

**Jan Jakubův**
Czech Technical University in Prague and University of Innsbruck

**Cezary Kaliszyk**
University of Innsbruck

**Miroslav Olšák**
Institut des Hautes Études Scientifiques

**Jelle Piepenbrock**
Czech Technical University in Prague and Radboud University

**Josef Urban**
Czech Technical University in Prague

**Abstract**

We significantly improve the performance of the E automated theorem prover on the Isabelle Sledgehammer problems by combining learning and theorem proving in several ways. In particular, we develop targeted versions of the ENIGMA guidance for the Isabelle problems, targeted versions of neural premise selection, and targeted strategies for E. The methods are trained in several iterations over hundreds of thousands untyped and typed first-order problems extracted from Isabelle. Our final best single-strategy ENIGMA and premise selection system improves the best previous version of E by 25.3% in 15 seconds, outperforming also all other previous ATP and SMT systems.

**2012 ACM Subject Classification** Theory of computation → Automated reasoning

**Keywords and phrases** E Prover, ENIGMA, Premise Selection, Isabelle/Sledgehammer

**Funding** This work was partially supported by the European Regional Development Fund under the Czech project AI&Reasoning no. CZ.02.1.01/0.0/0.0/15_003/0000466 (ZG, JP, JU), the ERC Starting Grant *SMART* no. 714034 (JJ, CK), Amazon Research Awards (JP, JU) and by the Czech MEYS under the ERC CZ project *POSTMAN* no. LL1902 (JJ, JP).

## 1 Introduction

Formal verification in interactive theorem provers (ITPs) increasingly benefits from general proof automation in the form of *hammers* [7] and guided tactical provers [4, 13, 37]. In particular, the Sledgehammer system [8] for Isabelle is today perhaps the most widely used strong general proof automation system in ITP. In the recent years, machine learning and related AI methods for proof automation have also been significantly developed [49]. Such methods are relevant for hammers in at least three ways: (i) learning-based *premise selection* [2, 3, 12, 38, 40, 41] usually improves the heuristic filters used by the hammers, (ii) learning-based *internal guidance* of the automated theorem provers (ATPs) used for the heavy lifting in the hammers usually improves on heuristic guidance of ATPs [16, 23, 25, 27, 28, 30, 42, 50], and (iii) targeted theorem proving strategies developed by automated strategy invention systems often improve on manually designed ATP strategies [21, 24, 43, 48].

Most recent versions of such AI/TP methods have been developed mainly on a fixed Mizar/MPTP corpus [29], to allow easy comparisons with previously developed methods. In particular, there the strongest 3-phase single-strategy version of the ENIGMA system (based on E [44, 45]) proves 56.35% of the holdout (test) toplevel theorems in 30s when using human-selected premises [16]. In higher time limits and by combining human and learning-based premise selection, ENIGMA and Vampire [33] today prove 75% of the toplevel Mizar theorems.[1] These are good reasons for transferring the methods to other ITP hammers.

A direct motivation for developing such AI/TP methods for Isabelle was a recent request from the Sledgehammer developers for an optimized version of ENIGMA for their GRUNGE-style [9] evaluation of multiple ATP systems and formats [11]. While it was not possible to do the work described in this paper on a two-week's notice, it prompted us to start exporting and analyzing the Isabelle datasets and developing suitable methods and systems for them.

## 1.1 Contributions

We significantly improve the performance of the E automated theorem prover on the Isabelle Sledgehammer problems by combining learning and theorem proving in several ways. First, in Section 2 we extract two large datasets of untyped first-order (FOF) and many-sorted first-order (TFF, TF0) Isabelle Sledgehammer problems, using the Isabelle tool Mirabelle. This results in almost 300000 aligned problems in each of the exports, spanning in total 1902 Isabelle theory files and covering a large number of topics in mathematics and formal verification. To our knowledge, these are so far the largest corpora of Isabelle Sledgehammer problems available today for training and evaluation of AI/TP systems. Section 2.1 analyzes the corpora, showing that they significantly differ from other large AI/TP datasets such as the Mizar/MPTP toplevel theorems [29] and the HOL4/GRUNGE toplevel theorems [9].

In Section 3, we find optimized E strategies and parameters for the corpora, which already improve on standard E on the problems. They are suitable also for combinations with the ENIGMA guidance, which is introduced in Section 4. We also describe there several extensions to ENIGMA that were developed to handle the Isabelle untyped and typed problems. Section 5 discusses the neural premise selection that we use and its extensions for the typed Isabelle setting. Section 6 evaluates the methods in several loops interleaving proving and learning from the proofs. Our ultimate performance results are: (i) improving in 15s the original E auto-schedule with the MePo filter by 25.3%, when using a single ENIGMA strategy with the best neural predictor, (ii) considerably improving over all other ATPs and SMTs by a single ENIGMA strategy combined with the best neural predictor, (iii) improving the performance of all other systems by using the neural predictor, and (iv) outperforming with ENIGMA all other ATPs and SMTs even when they are combined with our predictor.

## 2 Isabelle Problems

To train and evaluate the Isabelle ENIGMA, we need a dataset of Sledgehammer problems, which correspond to the proof obligations that users encounter when using Isabelle as an interactive prover. We decided to focus on all proof-intermediate goals visible to the users. This task has been tried as early as in the first versions of the MPTP system [47]. In Isabelle, it has been known as the "Judgement Day" evaluation, based on the paper with that title [8]. We have used the Isabelle/Mirabelle infrastructure to export all the problems encountered when building 179 Isabelle sessions. These sessions originate from 75 sessions distributed with Isabelle 2021-1, 80 selected sessions from the AFP [6], as well as 24 sessions distributed as part of IsaFoR [46]. All the sessions include in total 1902 Isabelle theory files. The sessions with most problems can be categorized as Analysis, Algebra, Java Semantics, Category

[1] https://github.com/ai4reason/ATP_Proofs/blob/master/75percent_announce.md

Theory, Protocols, Term Rewriting, and Probability Theory with the largest 26 sessions listed in Table 1.

| | | | |
|---|---|---|---|
| HOL-Nonstandard-Analysis | 1699 | Groebner-Macaulay | 4227 |
| Category2 | 1776 | HOL-ODE-Numerics | 4422 |
| Poincare-Bendixson | 1983 | HOL-MicroJava | 5183 |
| HOL-Number-Theory | 2071 | HOL-Auth | 5304 |
| MonoidalCategory | 2238 | HOL-Complex-Analysis | 5489 |
| HOL-Cardinals | 2268 | Groebner-Bases | 5710 |
| Core-DOM | 2280 | HOL-Computational-Algebra | 6280 |
| HOL-IMP | 2324 | Jordan-Normal-Form | 6786 |
| HOL-Data-Structures | 2353 | Category3 | 6818 |
| Dirichlet-Series | 2435 | HOL-Probability | 6954 |
| Slicing | 2517 | HOL-Decision-Procs | 7103 |
| HOLCF | 2524 | CR | 7341 |
| Formal-SSA | 2899 | HOL-Bali | 7804 |
| HOL-UNITY | 2938 | HOL | 7818 |
| HOL-Homology | 3022 | Goedel-HFSet-Semanticless | 8697 |
| HOL-ex | 3047 | HOL-Algebra | 9674 |
| CTRS | 3328 | HRB-Slicing | 10052 |
| HOL-Hoare-Parallel | 3733 | Jinja | 11520 |
| Signature-Groebner | 3762 | HOL-Library | 15627 |
| Valuation | 3786 | Bicategory | 16965 |
| Ordinary-Differential-Equations | 3885 | HOL-Nominal-Examples | 17145 |
| Smith-Normal-Form | 4045 | Group-Ring-Module | 19718 |
| Differential-Dynamic-Logic | 4158 | HOL-Analysis | 44172 |

**Table 1** The largest included sessions and their respective problem numbers

The Sledgehammer export allows multiple encodings of types, lambdas, and other options [5]. Since we are interested in the performance of learning-based first-order ATPs, we exported the problems in two first-order formats: TFF (also called TF0), i.e., many-sorted first-order logic, and FOF, i.e. untyped first-order logic. For all problems we pre-selected 512 relevant premises using the heuristic MePo filter [36] before the translation. This slightly overshoots the best performance (256 premises) obtained by most of the top systems[2] on the FOF and TFF problems in the recent Sledgehammer evaluation [11]. We use 512 premises because the heuristic MePo filter is known to be weaker than state-of-the-art selection systems (possibly pruning out some good premises too early), and also because the 512-premise results of the best systems in [11] are nearly identical[3] to the 256-premise results.[4]

For the other parameters, for E and Vampire we used the ones corresponding to the slice selected when no-slicing is used for a particular prover. Additionally, when extracting the FOF problems, we used the parameters used for such a slice in first-order E in the previous Isabelle version. These parameters have been optimized by the Isabelle/Sledgehammer developers based on experiments described in previous papers, e.g., [5]. To ease comparison

[2] Vampire is an exception: in [11] it is best with 512 premises, likely due to its optimized SInE filter [20].

[3] In particular, CVC5 - the winner in [11] - is only 3.7% (2626/2533) stronger with 256 premises.

[4] We could have used also 1024 premises, however already with 512 premises the datasets are becoming very large, making also the training of the ML systems technically challenging.

with [11], we use the polymorphic *g*?? [11] encoding together with lambda-lifting [22] for FOF and the native monomorphic encoding with lambda-lifting for TFF0.

Since the Mirabelle export has occasional problems with some theories and encodings (theory compilation fails or does not terminate with a particular export), we initially get different numbers of problems for the FOF (293587) and TFF (386619) exports. To align the two exports, we remove the non-overlapping problems, thus obtaining 276363 problems both in FOF and TFF that correspond to each other. As usual in machine learning, we then divide this dataset into the *training*, *devel*opment (validation) and *holdout* (ultimate testing) parts. This is done by randomly shuffling the list of the problems and dividing the shuffled list 90:5:5. This means that we have 248727 problems to train our systems on, 13818 development problems for controlling the hyperparameters of the learning and building the best portfolios, and 13818 holdout problems on which the trained systems are ultimately evaluated. We also sometimes use a 13818-big subset of the training set (*small trains*). The total size of the FOF dataset is about 50G compressed by gzip to 5.4G, while for the TFF dataset it is about 90G, compressed by gzip to 7.7G. The complete datasets are publicly released at our accompanying repository.[5]

The translation of the Isabelle/HOL problems to TPTP does not preserve the names across the problems. The naming inconsistency can be as simple as the naturals being given the constant name `nat` or `nat2` in an encoded TPTP problem (this one happens because the projection int-to-nat is also called `nat` in Isabelle), depending on the order of defined constants in a given problem. Additionally, Isabelle mangles names as part of the encoding. For example in the basic theorem `List.distinct`, which states that an empty list is not equal to an applied list constructor, an instance of the empty list can look like `nil_Pr1308055047at_nat` for an empty list of products of pairs of naturals. This motivates our use of anonymous methods for ENIGMA and premise selection in this work (Section 4,5).

## 2.1 Differences to Related ITP/ATP Datasets

The FOF and TFF Isabelle exports we use are intended to be sound but generally sacrifice completeness to optimize ATP performance. The possible sources of incompleteness include:

- The heuristic premise filter [36] pre-selecting only a fixed number of premises that are generally not guaranteed to justify the conjecture in Isabelle.
- In the encodings, polymorphic types (such as `'a list`) are heuristically pre-instantiated (*monomorphized*) by ground types. This is an established optimization going back at least to Harrison's implementation of the MESON tactic [19] in HOL Light [18], which can be seen as a particular kind of an abstraction step when reasoning in large theories. Without a full abstraction-refinement loop [35], this is an obvious source of incompleteness, in a similar way as premise selection with a fixed premise limit.
- Limited treatment of higher-order constructs such as lambda abstraction, typically not fully encoded in the FOF and TFF problems. The encodings employ lambda-lifting, which is usually improving the ATP performance in practice, but is generally incomplete.

When developing new strategies, ATPs and premise selection methods, such optimizations may be premature, having different beneficial or adverse effects on the methods. In particular, in the experiments conducted by us, we detect small amount of incompleteness already with the baseline systems. For example, CVC5 reports 256 problems in the whole TFF dataset to

[5] https://github.com/ai4reason/isa_enigma_paper

be countersatisfiable. On the other hand, once a proof is found, it is typically comparatively easy to replay from the minimized set of premises by any ATP.

In this sense, the monomorphized Isabelle datasets considerably differ from other datasets used for large AI/TP experiments such as the toplevel theorems in the Mizar and HOL 4 libraries [9]. There, replaying the minimized proofs may still be quite hard for ATPs, and the exports are typically striving for completeness, fully delegating various abstraction-refinement methods such as monomorphization and premise selection to the AI/TP systems that may implement more complicated procedures for them.

We measure this in more detail by comparing the clausified premise-minimized ATP problems solved by Vampire and E on the Isabelle FOF dataset (88888 problems) and the Mizar dataset (113332 problems) using several metrics computed in Table 2. The table shows that the number of clauses per minimized problem is 3.5 times higher in Mizar. This may indicate the difference between the (generally harder) toplevel ITP problems and the intermediate goals. The most interesting difference is that about two thirds of the clauses in the Mizar problems contain variables, while in Isabelle this is only 44.4% of the clauses. Combined with the much higher number of clauses in the Mizar problems, this leads to 5.14 times more clauses with variables in the Mizar problems. For clauses with equality, this ratio is 3.92, i.e., also slightly higher than the ratio of the clauses. This means that the Isabelle problems are (after minimization) much more ground and non-equational, and thus likely much more amenable to instantiation-based methods than the Mizar problems. We confirm this by running iProver [32] on both sets of minimized problems using only its Inst-Gen calculus. In Mizar it solves 58% of the problems while in Isabelle 93%, i.e., 60% more.

**Table 2** Statistics of the Isabelle and Mizar clausified premise-minimized FOF problems solvable by E and Vampire. AC is the average number of clauses per problem, VC is the average number of clauses with variables per problem, EC is that for clauses with equality, iProver-10s is the number of problems solved by iProver limited to inst-gen calculus in 10s, and iProver-10s ratio is the ratio of that to the total number of problems.

| Dataset | Problems | AC | VC | EC | iProver-10s | iProver-10s ratio |
|---|---|---|---|---|---|---|
| Isabelle FOF | 88888 | 10.15 | 4.51 | 2.63 | 83015 | 0.93 |
| Mizar | 113332 | 35.55 | 23.16 | 10.31 | 65679 | 0.58 |
| Ratio Miz/Isa | | 3.50 | 5.14 | 3.92 | | 0.62 |

## 3 Strategy Optimization for E and ENIGMA

ATP *strategies* play a critical role when proving theorems. Their targeted invention, optimization, and construction of their portfolios (*schedules*) may significantly improve the performance of the ATPs in different domains. We have also found that some ATP strategies behave better in combination with learning-based guidance of the ATPs than others, and that it often seems preferable to use a single strategy to produce the training data for ENIGMA.[6]

Our initial goals are thus to (i) find a strong set of E strategies for the datasets, and in particular, to (ii) find a single strong E strategy that behaves well in combination with

[6] The use of single vs multiple strategies in combination with ENIGMA is not yet strongly experimentally explored. See, e.g., [15] for a recent related analysis.

the ENIGMA guidance. We start exploring this on the FOF dataset, evaluating our 550 BliStr/Tune [24, 48] strategies previously invented on the Mizar, Sledgehammer, HOL, AIM and TPTP problems. This is done in two rounds. In the first round, we run all the 553 strategies on a smaller sample of 500 randomly selected FOF problems solvable by Vampire's CASC mode in 30 seconds.[7] After that, the 76 most performing and orthogonal strategies from the first run are evaluated on a bigger sample of 2000 Vampire-solvable problems. This yields the following top 2 strategies in the greedy cover:

```
protokoll_X----_auto_sine13 :995
protocol_eprover_f171197f65f27d1ba69648a20c844832c84a5dd7 :198
```

The first strategy uses E's auto-mode with a strong SInE filter, selecting up to 100 premises. Unlike in the Mizar problems, the `hypos` parameter of SInE is used here, giving the same importance to the local assumptions (TPTP role `hypothesis`) as to the conjecture. We have confirmed that this performs better than SInE without the parameter on the problems. This leads us to construct the ENIGMA features differently for Isabelle problems in Section 4.

The second strategy in the greedy cover (`f1711`) is the one working best in the Mizar/MPTP setting, where it also performs well when combined with the ENIGMA guidance. It is however significantly weaker (921 vs 995 solved problems) than the first auto-mode strategy. We conjecture that this is because it does not use SInE. Adding a strong SInE filter (with the "hypos" parameter) indeed improves its performance to 1022 problems, making it the strongest E strategy on the problems. Since it is also well behaved with the ENIGMA guidance, we use it in all further experiments. The base strategy (`f1711`) without any SInE filter will be denoted as $\mathcal{B}_{\mathsf{base}}$, while the version with the SInE as $\mathcal{B}_{\mathsf{sine}}$. With the clausification changes explained next we obtain two more strategies $\mathcal{B}_{\mathsf{base3}}$ and $\mathcal{B}_{\mathsf{sine3}}$.

## 3.1 Clausification

Clausification can have a large influence on the operation and performance of ATPs. In a setting with many complicated formulas, naive clausification can lead to exponential blow-ups. State-of-the-art ATPs counter that by introducing definitions for subformulas. E's clausifier uses heuristic counting of the occurrences of each subformula to decide when to introduce a new definition. The default factor (called `definitional-cnf`, `dc` for short) for this used by E has been experimentally optimized to be 24 many years ago on the TPTP benchmark. This may be however suboptimal for newer large-theory corpora, especially in encodings with type guards. Also, a possible explanation for the relatively large improvement of E by the aggressive SInE filter is that the clausification explodes quite frequently on the unfiltered problems. We investigate this in several ways.

First, we simply try to clausify all FOF problems with the default E options and a timeout of 60s. This results in a gzipped total size of 21G, i.e. four times the size of the gzipped FOF problems. This is however without 28212 (10% of all) problems that fail to get clausified within 60s. This is a lot, because ITP hammers typically give the ATPs a timeout of 15-30s to solve the whole problem.

This leads us to an experiment with smaller values for the definitional-cnf (`dc`) parameter on a sample of 1000 training problems. We use a 60s timeout for the clausification, measure the total size of gzipped cnfs, and the number of files where the clausification timed out. The results are shown in Table 3. The `dc` value of 3 is the last one where there are no timeouts, but it already gives a 4-time blowup over `dc` $= 2$.

[7] We use here Vampire as a quick pre-filter for targeting the solvable problems by E strategies because in our preliminary experiments Vampire performed significantly better than E.

**Table 3** Influence of the `dc` values on the clausification timeouts and size of the clausal problems.

| definitional-cnf (`dc`) | 1 | 2 | 3 | 4 | 24 |
|---|---|---|---|---|---|
| clausifications timed out in 60s (out of 1000) | 0 | 0 | 0 | 51 | 125 |
| gzipped size of all clausified problems (MB) | 36 | 47 | 163 | 120 | 77 |

**Table 4** 15s $\mathcal{B}_{\mathsf{base}}$ runs with/out SInE with different `dc` values on 1000 sample problems.

| definitional-cnf (`dc`) | 1 | 2 | 3 | 4 | 24 |
|---|---|---|---|---|---|
| problems solved with SInE | 242 | 268 | 271 | 266 | 263 |
| problems solved without SInE | 219 | 251 | 243 | 241 | 218 |

Both more aggressive premise selection and more aggressive introduction of new definitions can be used to counter the clausification blowup on the Isabelle problems. Since the SInE filter is only heuristic and usually inferior to trained premise selection, we prefer more aggressive use of new definitions. To measure how much the two methods interact, we evaluate our chosen strategy $\mathcal{B}_{\mathsf{base}}$ with and without SInE and with various `dc` values in 15s on our sample of 1000 problems. The results are summarized in Table 4. They confirm that the two methods interact a lot. Setting $\mathtt{dc} = 2$ replaces a lot of the improvement obtained by SInE with the default $\mathtt{dc} = 24$. Since the SInE and non-SInE versions peak at $\mathtt{dc} = 3$ and $\mathtt{dc} = 2$ respectively, we experiment with these values of `dc` in our further experiments. We denote $\mathcal{B}_{\mathsf{base3}}$ and $\mathcal{B}_{\mathsf{sine3}}$ the strategies obtained from $\mathcal{B}_{\mathsf{base}}$ and $\mathcal{B}_{\mathsf{sine}}$ by setting $\mathtt{dc} = 3$.

## 4 ENIGMA for Isabelle

State-of-the-art automated theorem provers (ATP), such as E, Prover9, and Vampire [33], are based on the saturation loop paradigm and the *given clause algorithm* [39]. The input problem, in first-order logic (FOF), is translated into a refutationally equivalent set of clauses, and a search for contradiction is initiated. The ATP maintains two sets of clauses: *processed* (initially empty) and *unprocessed* (initially the input clauses). At each iteration, one unprocessed clause is selected (*given*), and all of the possible inferences with all the processed clauses are generated (typically using resolution, paramodulation, etc.), extending the unprocessed clause set. The selected clause is then moved to the processed clause set. Hence the invariant holds that all inferences among processed clauses have been computed.

The selection of the "right" given clause is known to be vital for the success of the proof search. The first ENIGMA systems [14, 25–27] successfully implemented various ways of machine learning guidance for the clause selection based on gradient boosting decision trees (GBDT). Next generation ENIGMA [10,23] abstracts from symbol names with anonymization methods and additionally employs graph neural network models (GNN) for clause selection. The latest ENIGMA [16] additionally implements clause filtering of generated clauses (*parental guidance*), and overcomes a slower speed of GNN models with amortizing evaluation server.

## 4.1 Model Training and Given Clause Guidance

The training of ENIGMA models is usually done in a training/evaluation loop. This general approach applies both to clause guidance and when filtering the generated clauses.

1. The training data $\mathcal{T}$ are gathered from a number of previous successful proof searches. From each proof search, the training data consists of clauses processed during the proof search, labeled by flags *positive* or *negative* depending on whether they appear in the final proof. These labeled clauses are translated to a suitable format for the underlying selection model (vectors for GBDT models, and tensors for GNNs).
2. Based on data $\mathcal{T}$, a GBDT (or a GNN model) $\mathcal{M}$ is trained. This model is capable of recognizing *positive* clauses from *negatives* by assigning a score to an arbitrary clause.
3. The model $\mathcal{M}$ can be combined with an ordinary E's strategy $\mathcal{S}$ in a *cooperative* way, yielding the ENIGMA strategy $\mathcal{S} \oplus \mathcal{M}$. The ENIGMA strategy $\mathcal{S} \oplus \mathcal{M}$ uses the model $\mathcal{M}$ to guide the given clause selection inside E, and it inherits other behaviour from $\mathcal{S}$. In the cooperative setting, about 50% of the given clauses are selected as suggested by $\mathcal{M}$, while the remaining clauses are selected by the standard clause selection mechanism inherited from $\mathcal{S}$. Thus, ENIGMA compensates for a possible mistaken predictions of $\mathcal{M}$.
4. With new training data from new strategies, this process can be iterated.

## 4.2 Parental Guidance and Generated Clause Filtering

ENIGMA models are applied within E in two capacities: (1) given clause selection and (2) parental guidance for filtering of the generated clauses. Clausal parental guidance evaluates a new clause $C$ based only on the features of the parents of $C$. Parental guidance thus serves as a fast rejection filter: generated clauses with scores below a chosen threshold are put into the *freezer* set and are only revived if E runs out of unprocessed clauses. Furthermore, such frozen clauses are never evaluated by other (possibly more expensive) heuristics. This mechanism thus effectively (and in a complete way) curbs the typically quadratic growth of the set of generated clauses. Full details can be found in previous work [16] where it was found that the the parental guidance is most effective when the concatenated feature vectors of the parents are used as an input to the machine learning model. The data for training parental guidance is generated by classifying parents of proof clauses as *positive* and all other generated clauses during a proof search as *negative*. To balance the data, the ratio of negative to positive examples is a valuable hyperparameter.

## 4.3 Experiments with ENIGMA

ENIGMA was so far used only with first-order logic (FOF) data in the TPTP format. In this work, we extend the usability of ENIGMA models also to simply typed first-order formulas (TFF) of the TPTP format. In the case of GBDTs models, we simply forget the type annotations. Because GBDT ENIGMA models perform symbol name anonymization by replacing symbol names by their arities, all the simple type names would get translated to the same name anyway. In the case of GNN models, we embed the type information in the clause graphs by giving nodes representing variables of the same type by the same trained numerical representation (see Section 5).

ENIGMA models embed information about the conjecture being proved inside clause vectors/tensors. In this way, ENIGMA provides conjecture-specific suggestions. The conjectures are marked in the input format with the TPTP role `conjecture`. In these experiments, we additionally treat clauses with the TPTP role `hypothesis` just like conjectures. This helps to further differentiate among various Isabelle problems.

In this work, we use ENIGMA GBDT models for clause guidance inside E (for given clause selection and filtering of generated clauses), and we use the GNN models only for the task of premise selection. Section 5 describes how the GNN models are used for premise selection. The experimental evaluation described in Section 6 presents the results of training ENIGMA models for clause selection and parental guidance.

## 5 Premise Selection for Isabelle via Graph Neural Networks

A number of learning-based premise selection methods have been developed for large ITP corpora and hammers in the last two decades. See [2, 7, 34, 49] for their overviews. In a large evaluation done over the Mizar corpus,[8] the strongest method turned out to be a property-invariant graph neural network (GNN) based on the architecture previously used in several settings [23, 38, 51]. We use this algorithm also for the Sledgehammer problems here.

GNNs, and in particular this architecture preserve several invariants of theorem proving data, such as insensitivity to clause ordering and literal ordering. The inference (decisions) about which premises are relevant for a conjecture are based on several rounds of neural message passing in a special graph constructed from the clauses corresponding to the formulas. The property invariant architecture also strives to be fully anonymous, in the sense that it is invariant to all symbol names: the representations of symbols are only based on their connectivity with other elements in the formula. It also has a specific encoding for argument order that allows the network to partially preserve this information and it has a special handling of negation: terms of opposite polarity are related by the corresponding operation $*-1$ in the float based representation of the network.

This set of properties allows the architecture to perform well in various theorem-proving settings. On our Isabelle datasets, the symbol and name anonymity of the GNN is particularly important. As mentioned in Section 2, the symbol names and the formula names are not used consistently here, which would make the use of non-anonymous premise selection methods difficult. In this work, a 10-layer GNN was used. The sizes of the first layer embeddings were 4, 1, 4 for the *term*, *symbol* and *clause* nodes respectively. For the rest of the layers, the term, symbol and clause nodes were represented by vectors of size 32, 64 and 32 respectively. The last, non-message passing layer that has the task of predicting a probability for each premise had 128 neurons.

The GNN was newly modified to parse and make use of the typed TFF input. To take advantage of the type information, we train separate embeddings for all types (2539 in Section 6.4) that occur more than 10.000 times in the data. The GNN uses this type embedding when reading in a variable, and the type embedding can contain information about the type of the variable. Here, for simplicity, we chose to directly learn the embeddings (initial GNN values before the start of the message passing) for the typed variable nodes. This however does not fully preserve the anonymity of the symbols in the GNN, which is one the core design principles of this neural architecture. Adding instead an extra node in the GNN for each type would allow us to preserve the anonymity also for types. In this setting the GNN would learn to understand the types based only on their use in the current problem, possibly thus generalizing better. This approach is however more complicated than our current solution and is left as future work here.

[8] https://github.com/ai4reason/ATP_Proofs

The Isabelle problems are big and their clausification by our GNN parser may result in graphs with many clauses, even when we heuristically pre-reduce the initial set of formulas proposed by the MePo filter. This poses problems with the GPU memory (32 GB on our machines) both during training the GNN and when using it for predicting the relevant clauses. To counter that, we have introduced several limits related to the number of nodes in the clause graphs that allow us to skip very large clausified problems. The limit that we currently use skips any problem that contains more than 50000 term nodes after clausification (this corresponds roughly to the 95th percentile for the amount of term nodes in the problems).

## 6 Evaluation

We experiment with four variants of Isabelle problems. The first two are (1) FOF and (2) TFF without premise selection. Then there are two versions result from the GNN premise selector applied to the TFF data: (3) $\text{PRE}_1$ and (4) $\text{PRE}_2$.

First, Section 6.1 describes experiments with given clause guidance, and Section 6.2 describes experiments with adding parental guidance. These two experiments were partially used to obtain the training data for premise selection described in Section 6.3 and Section 6.4.

### 6.1 Evaluation of ENIGMA Given Clause Guidance

We perform three separate evaluations of the GBDT (LightGBM [31]) ENIGMA clause selection models on three different presentations of Isabelle problems. (1) On the FOF translation (without premise selection) in Section 6.1.1, (2) on the TFF translation (without premise selection) in Section 6.1.2, and (3) on the TFF translation with GNN premise selection in Section 6.1.3. The second premise selection dataset $\text{PRE}_2$ is not used here.

We experiment with combining training samples from different strategies. Different E strategies might use different term orderings affecting the clause normalization. Since the ENIGMA models are syntax based, we only combine training samples from *compatible* strategies, which perform equivalent clause normalization. At this point, we consider strategies to be *compatible* when they use the same term ordering and literal selection function.

#### 6.1.1 Experiment FOF: First-Order Translation

**Setup.** First, we experiment with the FOF translations of Isabelle problems without any premise selection method applied. E supports *sine* filters to reduce the number of axioms of large problems. Since the problems have no premise selection applied, we use two versions of the E strategy to obtain training problems: $\mathcal{B}_{\mathsf{sine}}$ uses a manually selected sine filter[9] and $\mathcal{B}_{\mathsf{base}}$ does not use a sine filter. We perform three training/evaluation loops as follows.

1. *Initial training data* $\mathcal{T}_0$: Evaluation of $\mathcal{B}_{\mathsf{base}}$ and $\mathcal{B}_{\mathsf{sine}}$ on the training problems.
2. Train the model $\mathcal{L}$ on the current data $\mathcal{T}$.
3. Evaluate $\mathcal{B}_{\mathsf{base}} \oplus \mathcal{L}$ and $\mathcal{B}_{\mathsf{sine}} \oplus \mathcal{L}$ on the training problems.
4. Extend data $\mathcal{T}$ and continue with step 2.

We combine the two base strategies with model $\mathcal{L}$ in a cooperative way. With model $\mathcal{L}$ we obtain two strategies with ENIGMA guidance, that is, $\mathcal{B}_{\mathsf{base}} \oplus \mathcal{L}$ and $\mathcal{B}_{\mathsf{sine}} \oplus \mathcal{L}$.

**Learning Statistics.** Table 5 presents training data statistics and models evaluation for the three training/evaluation loops performed in this FOF experiments. There is:

[9] `-sine='GSinE(CountFormulas,hypos,1.1„03,20000,1.0)'`

| | **notation** | | **training** | | | | **accuracy**[%] | | | **model** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *l* | *trains* | *model* | *probs* | *proofs* | *rows* | *filesize* | *acc* | *pos* | *neg* | *time* | *filesize* |
| 0 | $\mathcal{T}_0^{\rm FOF}$ | $\mathcal{L}_0^{\rm FOF}$ | 70K | 114K | 8M | 1.1G | 92.8 | 89.8 | 93.4 | 0:12 | 54.8M |
| 1 | $\mathcal{T}_1^{\rm FOF}$ | $\mathcal{L}_1^{\rm FOF}$ | 81K | 255K | 16M | 2.3G | 87.8 | 82.1 | 89.0 | 0:20 | 54.9M |
| 2 | $\mathcal{T}_2^{\rm FOF}$ | $\mathcal{L}_2^{\rm FOF}$ | 84K | 400K | 23M | 3.2G | 85.6 | 81.9 | 86.5 | 0:31 | 55.1M |

**Table 5** Experiment FOF: Learning statistics (Section 6.1.1).

- **training**: The column *probs* is the number of training problems in the training data, while the column *proofs* is the number of different successful proof runs, where we can have multiple proofs for a single problem. The column *rows* signifies the number of vectors in the training data, each vector corresponding to one clause in the proofs. The column *filesize* is the file size of the *compressed* training samples.
- **accuracy**: Columns *acc*, *pos*, *neg* show testing accuracies of each model on the testing set in percents. Column *acc* show the overall model accuracy, while columns *pos* and *neg* show testing accuracy on positive and negative testing samples separately.
- **model**: The column *time* shows the time needed for model training (in hours and minutes), and the column *size* shows the LightGBM model file size. Model file size is an important suggestion of the model ATP performance, since the model size influences the model loading time and prediction times in E.

When training a model, we set aside 5% of the training data in order to compute the testing **accuracy**. The model is trained on the remaining 95%. [10] This split is done on the level of solved problem names rather than on proofs or vectors so that all the proofs of a single problem will appear either in the 95% training subset, or all in the 5% testing subset. This is important to keep the testing set unbiased. Otherwise, the testing data can partially overlap with the training data, since two proofs of the same problem tend to be quite similar. This split on solved problem names is computed independently in every loop iteration. This split is done only on the training problems of the global training/development/holdout split used for further experiment in this paper.

From the numbers in Table 5, we can see that the number of solved problems (column *probs*) in the data increases with every loop iteration but much more slowly than the value in the column *proofs*. This means that we are obtaining duplicate proofs for already solved problems, since we include all the proofs for all solved problems in the training data in this experiment. Note that the testing accuracies decrease with increasing training data size. All the models have been built in less than 30 minutes and result in a similarly sized model file. Also note that number of *proofs* grows much faster than the problems solved (*probs*). It shows that we often prove the same problems.

**ATP Evaluation.** Table 6 shows the ENIGMA models performance separately on training (**trains**) and on development problems (**devel**). Since the development problems were not used during the training in any way, this evaluation tells how much are the ENIGMA model over-fitting on the training files.

[10] The numbers in the **training** columns are only on the training 95% subset.

| | **strategy** | | **trains solved by** | | | | **devels solved by** | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $l$ | *base* | *sine* | *base* | *sine* | *both* | *total* | *base* | *sine* | *both* | *total* |
| - | $\mathcal{S}_{\mathsf{base}}$ | $\mathcal{S}_{\mathsf{size}}$ | **56 921** | **65 124** | *75 080* | *75 080* | **3114** | **3567** | *4084* | *4084* |
| 0 | $\mathcal{S}_\star \oplus \mathcal{L}_0^{\mathrm{FOF}}$ | | **77 084** | **72 869** | *85 903* | *86 661* | **3888** | **3886** | *4552* | *4784* |
| 1 | $\mathcal{S}_\star \oplus \mathcal{L}_1^{\mathrm{FOF}}$ | | **80 613** | **74 191** | *87 734* | *89 886* | **3933** | **3851** | *4516* | *4947* |
| 2 | $\mathcal{S}_\star \oplus \mathcal{L}_2^{\mathrm{FOF}}$ | | **81 640** | **74 878** | *88 566* | *91 261* | **3963** | **3894** | *4558* | *5036* |

**Table 6** Experiment FOF: ATP performance (Section 6.1.1).

Every row describes the performance of two strategies specified in the column **strategy**. Problems solved by the two strategies individually are in the first two **bold** columns. *Italics* values display a total cover of set of strategies. The column *both* shows the number of problems solved both by the two strategies together. This is helpful to estimate the complementarity of *base* and *sine* strategies. Two strategies are *complementary*, when they solve different problems. The column *total* shows the cumulative number of problems solved by all the current strategies (above in the table).

In Table 6, we see that the *sine* strategy performed better than *base* initially. However, from the first learning the *base* strategy dominates. This suggests that ENIGMA learns to do premise selection on its own to some extent (when trained on the samples from the *sine* strategy). All *base* and *sine* strategies are, however, quite complementary. In total, we start with 75 080 solved problems and we end up with 91 261 after the learning, almost 22% improvement on trains (23% on devels). The best single strategy is improved by 25% on trains (and by 11% on devels).

It is interesting to observe, how the *base* strategies in one iteration improves on both *base* and *sine* from the previous iteration, as if merging the two strategies into one. It suggests that additional proof samples from compatible but complementary strategies could lead to an additional improvement. We further investigate this in the next experiment (Section 6.1.2).

### 6.1.2 Experiment TFF: Typed First-Order Formulae

**Setup.** We perform a similar experiment as for the FOF, but this time targeted to the TFF Isabelle translation.

1. Again, we start with the training data obtained by the evaluation of $\mathcal{B}_{\mathsf{base}}$ and $\mathcal{B}_{\mathsf{sine}}$.
2. We run three iterations of the training/evaluation loop.
3. After the three iterations, we additionally evaluate two more pure E strategies $\mathcal{B}_{\mathsf{base3}}$ and $\mathcal{B}_{\mathsf{sine3}}$ which improve on $\mathcal{B}_{\mathsf{base}}$ and $\mathcal{B}_{\mathsf{sine}}$ by adjusting E's clausification algorithm (switch E's option "`definitional-cnf`" from 24 to 3).
4. We perform two more training/evaluation loops with the expanded training data.

**Learning Statistics.** Table 7 presents the machine learning evaluation (in the same format as Table 5 described in Section 6.1.1). Before the fourth loop ($l = 3$), we additionally evaluate all the strategies $\mathcal{S} \oplus \mathcal{L}$, for $\mathcal{S}$ ranging over $\mathcal{B}_{\mathsf{base3}}$ and $\mathcal{B}_{\mathsf{sine3}}$, and for $\mathcal{L}$ ranging over the models of the first three loops. This gives us additional training data for the fourth iteration, reflected in the table by a sudden increase in both solved problems (*probs*) and *proof* count (in the row $l = 3$). We see similar training times and model sizes as in the FOF experiment.

| | notation | | training | | | | accuracy[%] | | | model | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *l* | *trains* | *model* | *probs* | *proofs* | *rows* | *size* | *acc* | *pos* | *neg* | *time* | *size* |
| 0 | $\mathcal{T}_0^{\mathrm{TFF}}$ | $\mathcal{L}_0^{\mathrm{TFF}}$ | 108K | 186K | 10,3M | 1.2G | 89.6 | 86.2 | 90.2 | 12:36 | 54.8M |
| 1 | $\mathcal{T}_1^{\mathrm{TFF}}$ | $\mathcal{L}_1^{\mathrm{TFF}}$ | 114K | 383K | 19,6M | 2.2G | 85.0 | 78.8 | 86.2 | 20:29 | 55.0M |
| 2 | $\mathcal{T}_2^{\mathrm{TFF}}$ | $\mathcal{L}_2^{\mathrm{TFF}}$ | 117K | 587K | 27,9M | 3.1G | 82.6 | 77.4 | 83.8 | 20:52 | 55.1M |
| 3 | $\mathcal{T}_3^{\mathrm{TFF}}$ | $\mathcal{L}_3^{\mathrm{TFF}}$ | 122K | 822K | 39,3M | 4.3G | 81.4 | 77.8 | 82.2 | 23:17 | 55.2M |
| 4 | $\mathcal{T}_4^{\mathrm{TFF}}$ | $\mathcal{L}_4^{\mathrm{TFF}}$ | 123K | 1.03M | 48,6M | 5.3G | 80.9 | 77.6 | 81.7 | 29:46 | 55.3M |

**Table 7** Experiment TFF: Learning statistics (Section 6.1.2).

| | strategy | | trains solved by | | | | devels solved by | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *l* | *base* | *sine* | *base* | *sine* | *both* | *total* | *base* | *sine* | *both* | *total* |
| - | $\mathcal{S}_{\mathsf{base}}$ | $\mathcal{S}_{\mathsf{size}}$ | **100 259** | **98 317** | *114 838* | *114 838* | **5532** | **5403** | *6347* | *6347* |
| 0 | $\mathcal{S}_\star \oplus \mathcal{L}_0^{\mathrm{TFF}}$ | | **108 377** | **101 271** | *118 262* | *121 353* | **5468** | **5347** | *6222* | *6692* |
| 1 | $\mathcal{S}_\star \oplus \mathcal{L}_1^{\mathrm{TFF}}$ | | **113 729** | **103 382** | *121 995* | *124 795* | **5788** | **5471** | *6466* | *6894* |
| 2 | $\mathcal{S}_\star \oplus \mathcal{L}_2^{\mathrm{TFF}}$ | | **115 790** | **104 270** | *123 400* | *126 344* | **5934** | **5505** | *6547* | *6894* |
| * | $\mathcal{S}_{\mathsf{base3}}$ | $\mathcal{S}_{\mathsf{sine3}}$ | **106 132** | **100 904** | *118 925* | *132 552* | **5881** | **5522** | *6515* | *7160* |
| 3 | $\mathcal{S}_{\star 3} \oplus \mathcal{L}_3^{\mathrm{TFF}}$ | | **122 492** | **107 035** | *127 955* | *133 222* | **6293** | **5673** | *6785* | *7280* |
| 4 | $\mathcal{S}_{\star 3} \oplus \mathcal{L}_4^{\mathrm{TFF}}$ | | **122 931** | **107 316** | *128 339* | *133 762* | **6277** | **5704** | *6812* | *7326* |

**Table 8** Experiment TFF: ATP performance (Section 6.1.2).

**ATP Evaluation.** Table 8 presents the ATP evaluation (in the same format as Table 6 described in Section 6.1.1). As opposed to the FOF experiment, the *base* strategies dominate from the beginning. Both strategies are still highly complementary. The evaluation of $\mathcal{B}_{\mathsf{base3}}$ and $\mathcal{B}_{\mathsf{sine3}}$ strategies boosts the number of solved trains from 126 344 to 132 552. This highly improves the performance of the best strategy (*base*) in the fourth iteration ($l = 3$) from 115 790 to 122 492, that is, by 5.8%. It shows that additional external training data can be quite useful during the training. We further investigate this issue in the next experiment (Section 6.1.3).

### 6.1.3 Experiment $\mathrm{PRE}_1$: First GNN Premise Selection

**Setup.** Here we experiment with GNN premise selection data $\mathrm{PRE}_1$ obtained by applying GNN premise selection to the TFF problems. The GNN premise selection produces several collections of the training problems (called *slices*) with a slightly different clause selection criterion. We experiment with two slices $\mathrm{PRE}_1^{-1}$ and $\mathrm{PRE}_1^{64}$, which were experimentally found well performing and complementary. Our first experiment is aimed at generating a large collection of training samples.

1. We perform three loops of training/evaluation, just as in the TFF experiment, separately on $\mathrm{PRE}_1^{-1}$ and $\mathrm{PRE}_1^{64}$. We loop with the base strategies $\mathcal{B}_{\mathsf{base3}}$ and $\mathcal{B}_{\mathsf{sine3}}$.

| **notation** | | **training** | | | | **accuracy[%]** | | | **model** | |
|---|---|---|---|---|---|---|---|---|---|---|
| *trains* | *model* | *probs* | *proofs* | *rows* | *size* | *acc* | *pos* | *neg* | *time* | *size* |
| $\mathcal{T}^{\mathrm{PRE}_1}_{\mathsf{three}}$ | $\mathcal{L}^{\mathrm{PRE}_1}_{\mathsf{three}}$ | 108K | 186K | 10,3M | 1.2G | 89.6 | 86.2 | 90.2 | 12:36 | 54.8M |
| $\mathcal{T}^{\mathrm{PRE}_1}_{\mathsf{six}}$ | $\mathcal{L}^{\mathrm{PRE}_1}_{\mathsf{six}}$ | 133K | 763K | 28,6M | 3.26G | 77.6 | 76.8 | 78.0 | 25:44 | 77.9M |

**Table 9** Experiment PRE$_1$: Learning statistics (Section 6.1.3).

| | **trains solved by** | | | | **devels solved by** | | | |
|---|---|---|---|---|---|---|---|---|
| *strategy* | $\mathrm{PRE}_1^{-1}$ | $\mathrm{PRE}_1^{64}$ | *both* | *total* | $\mathrm{PRE}_1^{-1}$ | $\mathrm{PRE}_1^{64}$ | *both* | *total* |
| $\mathcal{S} = \mathcal{S}_{\mathsf{base3}}$ | **122 196** | **117 341** | *126 323* | *126 323* | **6706** | **6462** | *6955* | *6955* |
| $\mathcal{S} \oplus \mathcal{L}^{\mathrm{PRE}_1}_{\mathsf{three}}$ | **127 606** | **120 248** | *129 971* | *132 431* | **6800** | **6495** | *6994* | *7251* |
| $\mathcal{S} \oplus \mathcal{L}^{\mathrm{PRE}_1}_{\mathsf{six}}$ | **132 063** | **123 229** | *134 544* | *135 823* | **6994** | **6591** | *7153* | *7380* |

**Table 10** Experiment PRE$_1$: ATP performance (Section 6.1.3).

2. We merge the training data from the previous two separate experiments and perform three more loops on the merged data. However, we drop the *sine* strategies and evaluate only the strategy $\mathcal{B}_{\mathsf{base3}} \oplus \mathcal{L}$ on the two PRE$_1$ slices.
3. From the above we collect a large database of 108K proved training problems. Since the collection can contain duplicate proofs of a single problem, we select just three proofs per problem. We use the proof pos/neg ratios as a measure of proof similarity, and select proofs thusly different.
4. The training data from the last step, denoted $\mathcal{T}^{\mathrm{PRE}_1}_{\mathsf{three}}$, gives us one final model $\mathcal{L}^{\mathrm{PRE}_1}_{\mathsf{three}}$.

Next experiment tries to gather even more training samples.

1. We additionally consider training data from the previous TFF experiments.
2. We gather even more valuable training samples from ENIGMA parental guidance experiments on slices $\mathrm{PRE}_1^{-1}$ and $\mathrm{PRE}_1^{64}$.
3. We select three proofs per problem from TFF samples.
4. We select three proofs per problem from PRE$_1$ samples.
5. The training data $\mathcal{T}^{\mathrm{PRE}_1}_{\mathsf{sixes}}$ contain six proofs per problem and yield model $\mathcal{L}^{\mathrm{PRE}_1}_{\mathsf{sixes}}$.

**Learning Statistics.** Table 9 presents the machine learning evaluation (in the same format as Table 5 described in Section 6.1.1). Note the huge difference in the number of *proofs*, resulting in much larger *training data size*. The second training data include proofs of more than 25K additional problems (*probs*). Training times and model sizes clearly reflect the training file size.

**ATP Evaluation.** Table 10 presents the ATP evaluation (in the same format as Table 6 described in Section 6.1.1). Here, however, we evaluate the single strategy $\mathcal{B}_{\mathsf{base3}} \oplus \mathcal{L}$ on slices $\mathrm{PRE}_1^{-1}$ and $\mathrm{PRE}_1^{64}$, instead of using two *base* and *sine* strategies.

Firstly, we note the effect of the premise selection itself. The performance on $\mathcal{B}_{\mathsf{sine3}}$ improved by more than 15% from the previous experiment (from 106 132 to 122 196). The model $\mathcal{L}^{\mathrm{PRE}_1}_{\mathsf{three}}$ performs quite well, being trained on proofs 108K problems, it solves almost

128K problems. The model $\mathcal{L}_{\mathsf{six}}^{\mathrm{PRE}_1}$ further boosts the performance, showing that combining of training data from various compatible sources might be beneficial. Comparing the performance on trains with the performance on devels, we can conclude that ENIGMA LightGBM clause selection models slightly overfit but they are still capable of generalization.

## 6.2 Evaluation of the Parental Guidance

**Setup.** Parental guidance models are co-trained with clause selection models in a series of loops over the training data. In each loop iteration, the LightGBM parameters for parental guidance models are tuned using a series of grid searches with Optuna [1]. These are the number of leaves, the bagging fraction and frequency, the minimum number of samples to create a new leaf, and L1 and L2 regularization. The learning rate is fixed at 0.15, the maximum tree depth is capped at 256 and the number of trees is 250. The number of leaves is varied between 256 and 3333. The best result of each grid search is used for the next parameter's grid search. Accuracy on positive training examples is considered twice as important as the accuracy on negatives when choosing which parameters perform best. There are multiple reasons for this. A primary reason is that the confidence in positive examples is higher than confidence for the classification of negatives because a negative clause in one successful proof search could be positive in another proof search. The resulting model is evaluated with the nine parental filtering thresholds, $\{0.03, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$, over a set of 300 problems from the development set for 30 seconds. This is done for the vanilla TFF problems as well as the problems with premise selection slices. The best run (as evaluated by a greedy cover) on each version of the problems is then run on the full training set. Then the problems from runs in the total greedy cover are used as data for the next iteration of looping. This means that some problems can have over 10 proofs in the training data.

**Iterations.** The training of parental guidance was done with the aim to develop as strong a performance as possible, using diverse data. The models for loops $\mathcal{L}_1$ and $\mathcal{L}_2$ are run and trained on the TFF data that do not use premise selection. Models $\mathcal{L}_3$ and $\mathcal{L}_4$ are only run on the small trains set. The models $\mathcal{L}_3$ to $\mathcal{L}_8$ are run on $\mathrm{PRE}_1$. Finally, models $\mathcal{L}_9$ and $\mathcal{L}_{10}$ are run on $\mathrm{PRE}_2^{-1}$. The largest performance jumps correspond to the addition of premise selection (Table 11). The strongest parental guidance models are always on the $\mathrm{PRE}^{-1}$ premise selection data and the $\mathrm{PRE}_1^{64}$ slices provide fewer complementary problems than the baseline TFF problems.

**The best model** $\mathcal{L}_{10}$, with the second premise selection slices, $\mathrm{PRE}_2^{-1}$, proves 168 problems (56%) in 30s on the parameter tuning development set, and 137 893 problems (55.4%) on the training set in 15s. In 30s, $\mathcal{L}_{10}$ proves 7472 problems (54.1%) on the development set and 7466 problems (54%) on the holdout. Without premise selection, $\mathcal{L}_{10}$ proves 133 390 training problems (53.6%), which indicates that training on the premise selection data transfers back to the original problems. The remaining results are presented in Table 11. This parental+ENIGMA model is our final product. It solves 7395 holdout problems in 15s, thus significantly improving over unguided E and also over all other ATPs and SMTs. It also outperforms all other ATPs and SMTs even when they use our best premises (Table 12).

## 6.3 First Training of Premise Selection on TFF Problems ($\mathrm{PRE}_1$)

We have done several large experiments with the GNN-based premise selection (Section 5), first in the untyped and then in the typed setting. For lack of space we include below only

| | $\mathcal{L}_1$ | $\mathcal{L}_2$ | $\mathcal{L}_3$ | $\mathcal{L}_4$ | $\mathcal{L}_5$ | $\mathcal{L}_6$ | $\mathcal{L}_7$ | $\mathcal{L}_8$ | $\mathcal{L}_9$ | $\mathcal{L}_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| small trains | 6475 | 6718 | 7081 | 7140 | 7312 | 7351 | 7407 | 7417 | 7647 | **7705** |
| devel | 6241 | 6462 | 6928 | 6892 | 6566 | 6850 | 7070 | 7115 | 7277 | **7379** |
| holdout | 6251 | 6459 | 6886 | 6843 | 6581 | 6816 | 7015 | 7062 | 7352 | **7395** |

**Table 11** Parental guidance iterations on small trains, devel, and holdout (13 818 problems in 15s). Loops $\mathcal{L}_1$ and $\mathcal{L}_2$ are run on TFF data, $\mathcal{L}_3$ to $\mathcal{L}_8$ on $\mathrm{PRE}_1^{-1}$, and $\mathcal{L}_9$ and $\mathcal{L}_{10}$ on $\mathrm{PRE}_2^{-1}$.

the two final experiments on the TFF data, where most of the ENIGMA runs were done.

For the first round of training the GNN on the TFF data we are using the proof data produced only by the base sine/nosine TFF runs of unguided E and the first three ENIGMA iterations on the TFF training set. Altogether these runs produce 1701284 proof dependencies. These dependencies are first deduplicated to 353875, and then we also for every problem $P$ remove all premise sets subsumed by a smaller premise set. This further decreases the size of the dataset to 242432 proof dependencies, for 131309 unique solved problems. Most of the solved problems (80993) have after this redundancy elimination only one solution, while for the remaining ones we get from 2 to 16 different solutions. Since problems with 1 to 3 solutions dominate the dataset (163650 out of the total 242432 solutions), we do not do further pruning of over-represented proofs for the training (as in the next training run).

For this first training and prediction we do not yet use the new typed extensions of the GNN. Instead, all TFF formulas are stripped of their type information and given to the network as untyped FOF. Each problem uses its original conjecture, the positives are the premises used in a given proof and the negatives are all other premises for that problem (i.e., all the MePo premises). This sometimes leads to large training inputs, so we normalize them to have size at most 500KB by randomly removing negatives. The whole training dataset has size 46GB. We then train the GNN on it with batch size 10, learning rate 0.005, and with balancing the loss on the positive and negative premises.

The training for two full epochs on an NVIDIA Volta 100 takes about 12 hours, saving the weights 16 times. The balanced accuracy increases from 0.8533 to the final 0.9067 (0.9061 vs 0.9073 on positives vs negatives) in our final snapshot, which we then use for prediction over all 276363 problems. This is parallelized over four GPUs, taking several hours. For each problem we use the predictions to produce 5 premise selections based on the GNN score threshold (1,0,-1,-2,-3,-4), and 5 premise selections based on old-style top slices of the ranked premises (16,32,64,128,256). We do a small search with 200 development problems and the base strategy over this grid, which is won by the -1-based predictions, best complemented by the 64-based predictions. These premise selections are denoted $\mathrm{PRE}_1^{-1}$ and $\mathrm{PRE}_1^{64}$ in the other parts of this paper. E/ENIGMA are then evaluated on both of them, while we also evaluate other systems only on the -1-based predictions (Table 12).

## 6.4 Second Training of Premise Selection on TFF Problems ($\mathrm{PRE}_2$)

The second premise selection training is done by the typed version of the GNN (Section 5), using explicitly 2539 types that occur with frequency higher than 10000 in the training data. The remaining types (over 300000 in the training set) are mapped to the same generic embedding, which means that the GNN treats them all as the same type. The overhead for the 2539 distinguished most frequent types increases the size of the GNN only by 100kb.

**Table 12** Final comparison with non-ENIGMA systems: E2.6 with its auto-schedule, CVC5, and Vampire-CASC (master 4909). Each run standalone (MePo predictions) and with the first/second -1 GNN TFF predictions. The last entry is the final/best $Loop_{10}$ (parental) ENIGMA (Section 6.2).

| method | E auto-sched. | CVC5 | Vampire | $\mathcal{L}_{10}$ ENIGMA |
|---|---|---|---|---|
| 15s devel, no premsel | 5891 | 7053 | 6452 | 7133 |
| 15s holdout, no premsel | 5903 | 7051 | 6454 | 7139 |
| 30s holdout, no premsel | 6089 | 7140 | 6945 | 7170 |
| 15s devel, preds -1 (1st round) | 6968 | 7211 | 7023 | 7191 |
| 15s holdout, preds -1 (1st round) | 6956 | 7158 | 6978 | 7155 |
| 15s devel, preds -1 (2nd round) | 7074 | 7394 | 7132 | 7379 |
| 15s holdout, preds -1 (2nd round) | 7066 | 7372 | 7118 | **7395** |
| 30s holdout, preds -1 (2nd round) | 7139 | 7398 | 7397 | **7466** |

The training uses again a batch size of 20 and a learning rate of 0.0005. The training dataset is created from all TFF training problems solved in the previous loops, both by E/ENIGMA and CVC5 and Vampire. This gives 823141 unique premise selections for 146576 solved problems. The 823141 unique premise selections are again minimized with respect to subsumption, reducing them to 488186 minimal premise selections. To address the imbalance caused by having various numbers of proofs for a single problem in the training set, we keep at most three proofs for each problem. This further reduces the set to 292080 examples. The examples are again all reduced to a size of at most 500KB.

This results in our final training set with an overall size of about 60GB. Since the reduction of the TFF inputs does not generally guarantee to prevent a blow-up during the clausification, we also further use here a size limit of 50000 nodes inside the GNN parser (Section 5) and filter out such large graphs which may otherwise deplete the GPU memory. The GNN is trained for full two epochs on the data, taking about one day on a single NVIDIA V100 GPU and storing the weight files 15 times per epoch. For producing the final predictions, we take the 28th weights with the highest balanced accuracy of 0.9221 (0.9391 / 0.9051 for positives/negatives). We produce the same grid of predictions as in the first round for all problems. The -1-based predictions are again the winner, best complemented by the 0-based predictions. These premise selections are denoted $\mathrm{PRE}_2^{-1}$ and $\mathrm{PRE}_2^{0}$ in the other parts of this paper. Table 12 shows that also all non-ENIGMA systems benefit from the GNN predictions, and that the second round improves over the first round of predictions for all of them.

## 7 Conclusion

We have developed versions of the ENIGMA systems and neural premise selectors for the Isabelle Sledgehammer problems. Our best single-strategy system using the parental ENIGMA guidance and the typed GNN premise selection solves 7395 holdout problems in 15s, improving on original E's auto-schedule performance (5903) by 25.3%. It also improves on all other ATPs and SMTs, both when used standalone and when used in conjunction with our best neural premise selection. To achieve this, we have produced large corpora of Isabelle problems for training and evaluation of the AI/TP methods, and developed new extensions of our systems, especially for the typed setting.

## References

1 Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

2 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.

3 Alexander A. Alemi, François Chollet, Niklas Eén, Geoffrey Irving, Christian Szegedy, and Josef Urban. DeepMath - deep sequence models for premise selection. In Daniel D. Lee, Masashi Sugiyama, Ulrike V. Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2235–2243, 2016.

4 Lasse Blaauwbroek, Josef Urban, and Herman Geuvers. The tactician - A seamless, interactive tactic learner and prover for coq. In *CICM*, volume 12236 of *Lecture Notes in Computer Science*, pages 271–277. Springer, 2020.

5 Jasmin Christian Blanchette, Sascha Böhme, Andrei Popescu, and Nicholas Smallbone. Encoding monomorphic and polymorphic types. In Nir Piterman and Scott A. Smolka, editors, *TACAS*, volume 7795 of *LNCS*, pages 493–507. Springer, 2013.

6 Jasmin Christian Blanchette, Max W. Haslbeck, Daniel Matichuk, and Tobias Nipkow. Mining the archive of formal proofs. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge, editors, *Intelligent Computer Mathematics - International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings*, volume 9150 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 2015.

7 Jasmin Christian Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards QED. *J. Formalized Reasoning*, 9(1):101–148, 2016.

8 Sascha Böhme and Tobias Nipkow. Sledgehammer: Judgement Day. In Jürgen Giesl and Reiner Hähnle, editors, *IJCAR*, volume 6173 of *LNCS*, pages 107–121. Springer, 2010.

9 Chad E. Brown, Thibault Gauthier, Cezary Kaliszyk, Geoff Sutcliffe, and Josef Urban. GRUNGE: A grand unified ATP challenge. In *CADE*, volume 11716 of *Lecture Notes in Computer Science*, pages 123–141. Springer, 2019.

10 Karel Chvalovský, Jan Jakubův, Martin Suda, and Josef Urban. ENIGMA-NG: efficient neural and gradient-boosted inference guidance for E. In Pascal Fontaine, editor, *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, volume 11716 of *Lecture Notes in Computer Science*, pages 197–215. Springer, 2019.

11 Martin Desharnais, Petar Vukmirović, Jasmin Blanchette, and Makarius Wenzel. Seventeen provers under the hammer, 2022. https://matryoshka-project.github.io/pubs/seventeen.pdf.

12 Michael Färber and Cezary Kaliszyk. Random forests for premise selection. In Carsten Lutz and Silvio Ranise, editors, *Frontiers of Combining Systems - 10th International Symposium, FroCoS 2015, Wroclaw, Poland, September 21-24, 2015. Proceedings*, volume 9322 of *Lecture Notes in Computer Science*, pages 325–340. Springer, 2015.

13 Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar, and Michael Norrish. Tactictoe: Learning to prove with tactics. *J. Autom. Reason.*, 65(2):257–286, 2021.

14 Zarathustra Goertzel, Jan Jakubův, and Josef Urban. Enigmawatch: Proofwatch meets ENIGMA. In Serenella Cerrito and Andrei Popescu, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - 28th International Conference, TABLEAUX 2019, London, UK, September 3-5, 2019, Proceedings*, volume 11714 of *Lecture Notes in Computer Science*, pages 374–388. Springer, 2019.

15 Zarathustra Amadeus Goertzel. Make E smart again (short paper). In *IJCAR (2)*, volume 12167 of *Lecture Notes in Computer Science*, pages 408–415. Springer, 2020.

16 Zarathustra Amadeus Goertzel, Karel Chvalovský, Jan Jakubuv, Miroslav Olsák, and Josef Urban. Fast and slow enigmas and parental guidance. In Boris Konev and Giles Reger, editors,

*Frontiers of Combining Systems - 13th International Symposium, FroCoS 2021, Birmingham, UK, September 8-10, 2021, Proceedings*, volume 12941 of *Lecture Notes in Computer Science*, pages 173–191. Springer, 2021.

17 Georg Gottlob, Geoff Sutcliffe, and Andrei Voronkov, editors. *Global Conference on Artificial Intelligence, GCAI 2015, Tbilisi, Georgia, October 16-19, 2015*, volume 36 of *EPiC Series in Computing*. EasyChair, 2015.

18 John Harrison. HOL Light: A tutorial introduction. In Mandayam K. Srivas and Albert John Camilleri, editors, *FMCAD*, volume 1166 of *LNCS*, pages 265–269. Springer, 1996.

19 John Harrison. Optimizing Proof Search in Model Elimination. In M. McRobbie and J.K. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction*, number 1104 in LNAI, pages 313–327. Springer, 1996.

20 Krystof Hoder and Andrei Voronkov. Sine qua non for large theory reasoning. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *CADE*, volume 6803 of *LNCS*, pages 299–314. Springer, 2011.

21 Edvard K. Holden and Konstantin Korovin. Heterogeneous heuristic optimisation and scheduling for first-order theorem proving. In *CICM*, volume 12833 of *Lecture Notes in Computer Science*, pages 107–123. Springer, 2021.

22 R. J. M. Hughes. Super-combinators a new implementation method for applicative languages. In *Proceedings of the 1982 ACM Symposium on LISP and Functional Programming*, LFP '82, page 1–10, New York, NY, USA, 1982. Association for Computing Machinery.

23 Jan Jakubův, Karel Chvalovský, Miroslav Olsák, Bartosz Piotrowski, Martin Suda, and Josef Urban. ENIGMA anonymous: Symbol-independent inference guiding machine (system description). In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II*, volume 12167 of *Lecture Notes in Computer Science*, pages 448–463. Springer, 2020.

24 Jan Jakubův and Josef Urban. BliStrTune: hierarchical invention of theorem proving strategies. In Yves Bertot and Viktor Vafeiadis, editors, *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France, January 16-17, 2017*, pages 43–52. ACM, 2017.

25 Jan Jakubův 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.

26 Jan Jakubův and Josef Urban. Enhancing ENIGMA given clause guidance. In Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef, editors, *Intelligent Computer Mathematics - 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings*, volume 11006 of *Lecture Notes in Computer Science*, pages 118–124. Springer, 2018.

27 Jan Jakubův and Josef Urban. Hammering Mizar by learning clause guidance. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving, ITP 2019, September 9-12, 2019, Portland, OR, USA*, volume 141 of *LIPIcs*, pages 34:1–34:8. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

28 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, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings*, volume 9450 of *Lecture Notes in Computer Science*, pages 88–96. Springer, 2015.

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

30 Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olsák. Reinforcement learning of theorem proving. In *Advances in Neural Information Processing Systems 31:*

*Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 8836–8847, 2018.

31 Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, pages 3146–3154, 2017.

32 Konstantin Korovin. iprover - an instantiation-based theorem prover for first-order logic (system description). 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 *Lecture Notes in Computer Science*, pages 292–298. Springer, 2008.

33 Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *CAV*, volume 8044 of *LNCS*, pages 1–35. Springer, 2013.

34 Daniel Kühlwein, Twan van Laarhoven, Evgeni Tsivtsivadze, Josef Urban, and Tom Heskes. Overview and evaluation of premise selection techniques for large theory mathematics. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *IJCAR*, volume 7364 of *LNCS*, pages 378–392. Springer, 2012.

35 Julio César López-Hernández and Konstantin Korovin. An abstraction-refinement framework for reasoning with large theories. In *IJCAR*, volume 10900 of *Lecture Notes in Computer Science*, pages 663–679. Springer, 2018.

36 Jia Meng and Lawrence C. Paulson. Lightweight relevance filtering for machine-generated resolution problems. *J. Applied Logic*, 7(1):41–57, 2009.

37 Yutaka Nagashima and Ramana Kumar. A proof strategy language and proof script generation for isabelle/hol. In *CADE*, volume 10395 of *Lecture Notes in Computer Science*, pages 528–545. Springer, 2017.

38 Miroslav Olsák, Cezary Kaliszyk, and Josef Urban. Property invariant embedding for automated reasoning. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 1395–1402. IOS Press, 2020.

39 Ross A. Overbeek. A new class of automated theorem-proving algorithms. *J. ACM*, 21(2):191–200, April 1974.

40 Bartosz Piotrowski and Josef Urban. ATPboost: Learning premise selection in binary setting with ATP feedback. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, volume 10900 of *Lecture Notes in Computer Science*, pages 566–574. Springer, 2018.

41 Bartosz Piotrowski and Josef Urban. Stateful premise selection by recurrent neural networks. In *LPAR*, volume 73 of *EPiC Series in Computing*, pages 409–422. EasyChair, 2020.

42 Michael Rawson and Giles Reger. lazycop: Lazy paramodulation meets neurally guided search. In *TABLEAUX*, volume 12842 of *Lecture Notes in Computer Science*, pages 187–199. Springer, 2021.

43 Simon Schäfer and Stephan Schulz. Breeding theorem proving heuristics with genetic algorithms. In Gottlob et al. [17], pages 263–274.

44 Stephan Schulz. System description: E 1.8. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *LPAR*, volume 8312 of *LNCS*, pages 735–743. Springer, 2013.

45 Stephan Schulz, Simon Cruanes, and Petar Vukmirovic. Faster, higher, stronger: E 2.3. In Pascal Fontaine, editor, *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, volume 11716 of *Lecture Notes in Computer Science*, pages 495–507. Springer, 2019.

46 René Thiemann and Christian Sternagel. Certification of termination proofs using ceta. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *TPHOLs*, volume 5674 of *Lecture Notes in Computer Science*, pages 452–468. Springer, 2009.

47 Josef Urban. MPTP - Motivation, Implementation, First Experiments. *J. Autom. Reasoning*, 33(3-4):319–339, 2004.

48 Josef Urban. BliStr: The Blind Strategymaker. In Gottlob et al. [17], pages 312–319.

49 Josef Urban. ERC project AI4Reason final scientific report, 2021. http://grid01.ciirc.cvut.cz/~mptp/ai4reason/PR_CORE_SCIENTIFIC_4.pdf.

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

51 Zsolt Zombori, Josef Urban, and Miroslav Olsák. The role of entropy in guiding a connection prover. In *TABLEAUX*, volume 12842 of *Lecture Notes in Computer Science*, pages 218–235. Springer, 2021.