Fuchs (1995) – heuristic learned to be chosen with Gen Algo

Denzinger and Schulz (1996) – inference control heuristics for equational deduction. Data from prev proofs, select equations that are likely to be used in new situations. 1st eval fn works by symbolic retrieval of generalized patterns from a kkn base, 2nd eval fn compiles the knowledge into abstract term evaluation trees. **Analyzed proof protocols** by representing knowledge about protocols'n'proofs

Denzinger et al. (1997) – case-based reasoning, similarity concept, cooperation concept, reactive planning; still 'learn' from previous successful proof attempts'

Fuchs (1998) – Learn search-guiding heuristics by employing features in a simple, yet effective manner. Features used to adapts a heuristic to a solved problem. Utilize heuristic profitably for related target problems. **Prediction of usefulness of a fact.**

SNoW Carlsonn et al. (1999) – learning program that can be used as a general purpose mulit-class classifier and is specifically tailored for large number of features. Sparse Network of Winnows (not a typo). Sparse network of sparse linear functions over a pre-defined or incrementally acquired feature space. Several update rules may be used – sparse variations of the Winnow update rule, the Perceptron or Naive Bayes. Multi class learner. Decisions either binary or continuous (confidence in [0, 1]).

Proof General Aspinall (2000) – tool for developing proofs with ITP. Interaction based around proof script (seq of commands sent to ITP). Provides UI.

Mizar proof advisor Urban (2004) – MPTP (Mizar Problems for Theorem Proving) is system described; translates MML into FOL for ATPs and for generating thm proving problems corresponding to Mizar Mathematical Library. Mizar proof advisor used for selecting suitable axioms from the large library for an arbirtrary problem. Feature based ML framework, symbols are the features that characterise formulas. They had 40k targets and about 7k features. **SNoW** learning architecture used mainly (NLP archit, designed for large num of feat and targets).

MizarMode Urban (2006) – Emacs authoring environment. Code-generating Code-Browsing Code-searching methods. Auto gen proof skeletons, semantic browsing of articles, structured viewing, proof advice using **machinee learning tools** like Mizar Proof Advisor.

Lightweight relevance filtering ... Meng and Paulson (2009) – relevance filtering methods, based on counting fn symbols in clauses. Signature based relevance filter. Not exactly ML?...

The use of Data-Mining... Duncan (2007) – evaluate the applicability of data-mining techniques for tactics from large corpuses of proofs. Data mine information to find occuring patterns. Patterns are then evolved into tactics. Variable Length Markov Models used to predict next proof step.

MaLARea Urban (2007) – simple metasystem iteratively combining deductive Automated Reasoning tools (now the E and the SPASS ATP systems) with a machine learning component (now the SNoW system used in the naive

Bayesian learning mode). Intended use – large theories, i.e. large num of problems which in a consistent fashion use many axioms, lemmas, thms, etc. The system works in cycles of thm proving followed by ML from successful proofs, using the learned information to prune the set of available axioms for the next cycle. MPTP challenge - 142/252. Learning could be stated as creating an assoc of some features of the conjecture with proving methods. Features – just symbols appearing in them. "Proving method" – ordering of all av axioms. Goal – given symbols, produces ordering of axioms, according to expected relevancy wrt the set of symbols. Sufficiently simple to implement and quite efficient in the first experiments with thm proving over Mizar library. Deduce, learn, loop implemented via growing axiom set and growing timelimit policy. First try to solve cheaply (min num axioms / most relevant, lowest timelimit). On success, learning performed on the newly available solution and axiom/time limit dropped to min values. No success – increase limits. Details in paper. MLMLMLMLML SNoW used in NB mode, bcs of speed. One training example contains all the symbols of a solved conjecture w/ names of axioms needed. A bayes network is trained. Easier to just relearn every time. Trained classifier is used to prune the axiom set for the next runs – we take all the unsolved conjectures and create a testing example from each by taking all its symbols. The classifier run on this printing (ordered) axioms. This is then used to select req num of axioms. Usage of previous results exists.

SRASS Sutcliffe and Puzis (2007) – selection determined by semantics of the axioms and conjecture, heuristically ordered by a syntactic relevance measure. Many problems more solved. At each iter the process looks for a model of selected axioms and the neg of the conjecture. If no model found, then the conjecture is consequence. Otherwise, then an unselected axiom that is false in the model is moved to the set of selected axioms. Newly selected axiom excludes the model from the models of the selected axioms and neg conj, eventually leading to a situation where there are no models of the selected axioms and the negated conjecture. Unselected axioms selected in decreasing order of usefuleness. Syntactic relevance score for usefulness. Direct relevance is ratio of how many predicates and/or functors the have in common to how many they have overall. Contextual direct relevance uses 'contextual intersection'.

MaLARea SG Urban et al. (2008) – combines model-based and learning based methods for automated reasoning in large theories. The implementation is based on MaLARea. Extended by taking into account semantic relevance of axioms, similar to SRASS. Combined system outperforms both. Three extensions to selection of axioms. 1. check for countersatisfiability in runs where this is probable. Allows for countersatisfiability precheck to detect more cases when more axioms need to be added. 2. Use models found when a problem is found to be countersatisfiable, as an additional criterion for computing axiom relevance. Need to efficiently evaluate formulae in the models. 3. Extend axiom specification using a logical criterion: the set of axioms should exclude as many known models of the negated conjecture as possible. **Weird combination. Review!**

MaLeCoP Urban et al. (2011) – TABLEAU CALCULUS YAY!!! While in

MaLARea learning-based axiom selection is done outside unmodified theorem provers, in MaLeCoP the learning-based selection is done inside the prover, and the interaction between learning of knowledge and its application can be much finer. The general design that we propose is as follows (see also Figure 1): The theorem prover (P) should have a sufficiently fast communication channel to a general advisor (A) that accepts queries (proof state descriptions) and training data (characterization of the proof state together with solutions and failures) from the prover, processes them, and replies to the prover (advising, e.g., which clauses to choose). The advisor A also talks to external system(s) (E). A translates the queries and information produced by P to the formalism used by a particular E, and translates Es guidance back to the formalism used by P. At suitable time, A also hands over the (suitably transformed) training data to E, so that E can update its knowledge of the world on which its advice is based. A is free to spawn/query as many instances/versions of Es as necessary, and A is responsible for managing the guidance provided by them. Particular instances of Es that we have in mind are learning systems, however we believe that the tableau setting is also suitable for linking of SMT solvers, computer algebra systems, and all kinds of other AI systems, probably in a more straightforward way than for the resolution-based systems. SNoW again...

Premise selection .. and kernel methods Alama et al. (2011) – This work develops learning-based premise selection in two ways. First,a newly available minimal dependency analysis of existing high-level formal mathematical proofs is used to build a large knowledge base of proof dependencies, providing precise data for ATP-based re-verification and for training premise selection algorithms. Second, a new machine learning algorithm for premise selection based on kernel methods is proposed and implemented (Section 4 gives details) 0/1 features if sth appears in conjecture. Looking for a classifier fn which, given a conjecture c, estimates how useful p is for proving c. (still the aproach with chosing the best some premises) Maths explained nicely.

Flyspeck Kaliszyk and Urban (2014) – Trained on Flyspeck proofs. (HOL Light is an ITP) The procedure implemented for HOLLight is currently a combination of the external, internal, learning, and non-learning premise selection approaches. This procedure assumes the common ITP situations of a large library of (also definitional) theorems T_i and their proofs P_i (for definitions the proof is empty). The proofs refer to other theorems giving rise to a partial ordering of thms etended into their total chrono order. Procedure:

1. characterisations of thms and proofs are extracted in a simple format
2. dependency data are obtained by running ATPs on the ATP problems created from the hollight deps, i.e. tms are re-proved. Preferred (it is smaller) data. Exported as 1
3. external premise selectors preprocess the thm characterizations and the proof deps. Multiple characterizations and proof dependencies may be used

4. when an new conjecture is stated in hollight its characterization is extracted and sent to the pretrained first stage premise selectors.
5. the first stage premise selectors work as rankers. For a given conjecture characterization they produce a ranking of the available theorems (premises) according to their (assumed) relevance for the conjecture.
6. The best ranked premises are used inside hollight to produce atp problems. Several thresholds on num of included premises are used, resulting in multiple versions of the ATP problems.
7. The ATPs are called on the problems. Some of the best ATPs run in a strategy-scheduling mode combining multiple strategies. Some of the strategies always use the SInE (i.e. local, second stage) premise selection (with different parameters) and some other strategies may to use SINE when the ATP problem is sufficiently large.

ML of Premise Selection
All the currently used first-stage premise selectors are machine learning algorithms trained in various ways on previous proofs. A number of machine learning algorithms can be experimented with today, and in particular kernel-based methods and ensemble methods have recently shown quite good performance on smaller datasets such as MPTP2078. Scaling hard on large corpus. So far this work uses mostly sparse implementation of a multiclass NB classifier (SNoW again...). Several other fast incremental learning algorithms were briefly tried – perceptron and winnow algos (SNoW) and custom k-NN. Only k-NN produced enough additional prediction power.

At a given point during the library development, the training data available to the machine learners are the proofs of the previously proved theorems in the library. A frequently used approach to training premise selection is to characterize each proof P_i of theorem T_i as a (multi)set of theorems $\{T_{i_1}, ...T_{i_m} | T_{i_j} usedinP_i\}$ The training example will consist of the input characterization (features) of T_i (features) and the output characterization of T_i (labels) will be the multi set $\{T_i\}$ and the previous set. Such training examples can be tuned in various ways. For example the output theorems may be further recursively expanded with their own dependencies,the input features could be expanded with the features of their definitions, various weighting schemes and similarity clusterings can be tried, etc. This is also mostly left to future general research in premise-selection learning. Once the machine learner is trained on a particular development state of the library, it is tested on the next theorem T in the chronological order. The input features are extracted from T and given to the trained learner which then answers with a ranking of the available theorems. This ranking is given to HOL Light, which uses it to produce ATP problems for T with varied numbers of the best-ranked premise (FOR THE PREVIOUS) custom implementation of the k-nearest neighbor (k-NN) machine-learning method, which computes for a new example (conjecture) the k nearest (in a given feature distance) previous examples and ranks premises by their frequency in these examples. (
END)

Stronger automation for Flyspeck... Kaliszyk and Urban (2013) – 2 complementary AI methods used to improve strength of ai/atp service. First, several schemes for frequency-based feature weighting are explored in combination with distance-weighted k-nearest-neighbor classifier. A smaller improvement is obtained by evolving targeted E prover strategies on two particular premise selections, using the Blind Strategymaker (BliStr) system. The simplest way how to measure the similarity of formulas to the new conjecture is to compute the overlap of their (sparse) feature vectors. Neglected by our first implementation is however the sensitivity of k-NN to feature frequencies. Inverse Document Fequency weighting.

MaSh Kühlwein et al. (2013) – Sledgehammer had relevance filter (syntactic similarity). Mash learns from succsessful proofs. Integrates easily. *Draws on recent research in the context of Mizar and HOL Light.* CUSTOM version of a weighted sparse naiveB algo, that is faster than the NB in SNoW. Maintains persistent state and supports incremental, nonmonotonic updates. The main technical difficulty is to perform the learning in a fast and robust way without interfering with other activities of the proof assistant. Power users can enhance the learning by letting external provers run for hours on libraries, searching for simpler proofs. A particularly strong filter, MeSh, is obtained by combining MePo (MEPO is paulsons thingie which selects based on num of relevant symbols) and MaSh. Implementations refines this in several ways – chained facts take absolutie priority, local facts are preferred to gloobal, first order facts preferred to hol ones; rare symbols are weighted more heavily; etc. Mepo tents to perform best on that contain some rare symbols, otherwise it discriminates poorly. There is also issue of starvation: the filter with its iterative expansion of the set of relevant symbols effectively performs a best first search and may ignore some relevant facts close to the root. Provers given ranked selected facts. Time limit and number of facts vary (the classic setting). Once a proof is found, Sledgehammer miinimizes it by invoking the prover repeatedly with subsets of the facts it refers to.

The ML engine

Default algorithm NB adapted to fact selection. Manipulates thm proving concepts in an abstract way. Handcrafted features. Sources of proofs – all facts in theories. Most interesting lemmas, those written by man. (see paper for math details)

MaLARea04 Kaliszyk et al. (2014) – seems like nothing new...

FEMaLeCoP (great names btw) Kaliszyk and Urban (2015a) – FEMaLeCoP is a connection tableau theorem prover based on leanCoP which uses efficient implementation of internal learning-based guidance for extension steps. Despite the fact that exhaustive use of such internal guidance can incur a significant slowdown of the raw inferencing process, FEMaLeCoP trained on related proofs can prove many problems that cannot be solved by leanCoP. Femalecop is the first ai/atp system convincingly demonstrating that guiding the internal inference algorithms of theorem provers by knowledge learned from previous proofs can significantly improve the performance of the provers. In the MaLeCoP(Machine Learning Connection Prover) experiment we have shown that in principle it is possible to significantly prune the internal search space of leanCoP. In this work, we devise much stronger learning-based guidance for connection tableau by developing an AI/ATP system where the learning-based guidance is an optimized and tightly integrated part of the core inferencing algorithm and data structures. The basis of FEMaLeCoP is the OCaml version of leanCoP. We advise the selection of clause for every tableau extension step. (see paper) We use a fast custom OCaml implementation of the naive Bayes algorithm that learns the association of the features of the proof states (see below) with the contrapositives that were used for the successful tableau extension steps in previous proofs. We characterize the proof state as a weighted vector of symbols and/or (possibly generalized) terms extracted from all the literals on the active path. (see details) An optional problem-specific data-filtering step is to use the k-nearest neighbor (k-NN) algorithm for further restriction of the relevant training data. If this is used, we first find the k solved problems whose conjectures are (in the feature metric) closest to the current conjecture, and extract the training examples only from such problems. (INTEGRATED LIKE AN INTERNAL SYSTEM)

Mizar40 Kaliszyk and Urban (2015b) – 40% of thms solved in the MML. To achieve that, a large suite of AI/ATP methods is employed and further developed. We implement the most useful methods efficiently, to scale them to the 150000 formulas in MML. The main body of our work thusconsists of the following steps: (DETAILS IN PAPER...). Nothing really interesting or 'grand'

A learning based fact selector... Blanchette et al. (2016) – MaSh reintroduction...

DeepMath Irving et al. (2016) – "We propose a two stage approach for this task that yields good results for the premise selection task on the Mizar corpus while avoiding the hand-engineered features of existing state-of-the-art models. To our knowledge, this is the first time deep learning has been applied to theorem proving on a large scale." 4 main contributions: 1) demonstration for the first time that neural network models are useful for aiding in large scale automated logical reasoning without the need for hand-engineered features 2) Comparison of various network architectures 3) A method of semantic-aware definition-embeddings for function symbols that improves the generalization of formulas with symbols occurring infrequently 4) Analysis showing that neural network based premise selection methods are complementary to those with hand-engineered features. Motivation from NLP (see paper)
OverviewOfApproach
Pairwise relevance – predicting the probability that a given axiom is useful for proving a given conjecture. The conjecture and axiom sequences are separately embedded into fixed length real vectors, then concatenated and passed to a third network with 2 fully connected layers and logistic loss. During training time, the two embedding networks and the joined predictor path are trained jointly. As discussed in section 3, we train our models on premise selection data generated by a combination of various methods, including k-nearest-neighbor search on hand-engineered similarity metrics. We start with a first stage of character-level models, and then build second and later stages of word-level models on

top of the results of earlier stages.

Character level models – We begin by avoiding special purpose engineering by treating formulas on the character-level using an 80 dimensional one-hot encoding of the character sequence. These sequences are passed to a weight shared network for variable length input. (see paper) Word-level models – The character-level models are limited to word and structure similarity within the axiom or conjecture being embedded. However, many of the symbols occurring in a formula are defined by formulas earlier in the corpus, and we can use the axiom-embeddings of those symbols to improve model performance.

HOLstep Kaliszyk et al. (2017) – In this paper, we introduce a new dataset based on Higher-Order Logic (HOL) proofs, for the purpose of developing new machine learning-based theorem-proving strategies. Reasons for hol given in paper and how dataset is extracted. Several possible tasks are outlined.

End-to-End differentiable proving Rocktäschel and Riedel (2017) – doesn't fit in the picture (nice paper otherwise)

Reinforcement Learning of Theorem Proving Kaliszyk et al. (2018) – Thm proving algo that uses practically no domain heuristics for guiding its connection-style proof search. Instead, it runs many Monte-Carlo simulations guided by reinforcement learning from previous proof attempts. We produce several versions of the prover, parameterized by different learning and guiding algorithms. In this work, we remove this requirement, since it basically means that all shorter proof candidates have to be tried before a longer proof is found.

bare prover Our bare prover is based on a previous reimplementation [23] of leanCoP in OCaml (mlCoP).Unlike the Prolog version,mlCoP uses an explicit stack for storing the full proof state (used for the ML part). We first modify mlCoP by removing iterative deepening, i.e., the traversal strategy that makes sure that shorter (shallower) tableaux are tested before deeper ones. Instead, the bare prover randomly chooses extension and reduction steps operating on the current goal, possibly going into arbitrary depth. Next, we add playouts and search node visit counts. A play out of length d is simply a sequence of d consecutive extension/reduction steps (inferences) from a given proof state(a tableau with a selected goal). Inferences thus correspond to actions and are similar to moves in games. We represent inferences as integers that encode the selected clause together with the literal that connected to the goal. Instead of running one potentially infinite playout, the bare prover can be instructed to play n playouts of length d. Each playout updates the counts for the search nodes that it visits. Search nodes are encoded as sequences of inferences starting at the empty tableau. A playout can also run without length restrictions until it visits a previously unexplored search node. The next modification to this simple setup are bigsteps done after b playouts. They correspond to moves that are chosen in games after many playouts. Similarly, instead of starting all playouts always from scratch (empty tableau) as above, we choose after the first b playouts a particular single inference (bigstep), resulting in a new bigstep tableau. The next b playouts will start with this tableau, followed by another bigstep, etc.

rlcop extends the bare prover with (i) Monte-Carlo tree search balancing exploration and exploitation using the UCT formula. To implement Monte-Carlo tree search, we maintain at each search node i the number of its visits n_i, the total reward w_i, and its prior probability p_i. This is the transition probability of the action (inference) that leads from i's parent node to i. If no policy learning is used, the prior probabilities are all equal to one. The total reward for a node is computed as a sum of the rewards of all nodes below that node. In the basic setting, the reward for a leaf node is 1 if the sequence of inference s results in a closed tableau, i.e., a proof of the conjecture. Otherwise it is 0. (actually is slightly different – see paper)

Great paper, but need to review some RL for presentation.

References

Alama, J., Kühlwein, D., Tsivtsivadze, E., Urban, J., and Heskes, T. (2011). Premise selection for mathematics by corpus analysis and kernel methods. *CoRR*, abs/1108.3446.

Aspinall, D. (2000). Proof General: A generic tool for proof development. In *Tools and Algorithms for Construction and Analysis of Systems, 6th International Conference, TACAS 2000, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings*, pages 38–42.

Blanchette, J. C., Greenaway, D., Kaliszyk, C., Kühlwein, D., and Urban, J. (2016). A learning-based fact selector for Isabelle/HOL. *J. Autom. Reasoning*, 57(3):219–244.

Carlsonn, A. J., Cumby, C. M., Rosen, J. L., and Roth, D. (1999). SNoW user guide. Technical report, University of Illinois.

Denzinger, J., Fuchs, M., and Fuchs, M. (1997). High performance atp systems by combining several ai methods. In *Proceedings of the 15th International Joint Conference on Artifical Intelligence - Volume 1*, IJCAI'97, pages 102–107, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Denzinger, J. and Schulz, S. (1996). Learning domain knowledge to improve theorem proving. In *Automated Deduction - CADE-13, 13th International Conference on Automated Deduction, New Brunswick, NJ, USA, July 30 - August 3, 1996, Proceedings*, pages 62–76.

Duncan, H. (2007). *The use of data-mining for the automatic formation of tactics*. PhD thesis, University of Edinburgh, UK.

Färber, M., Kaliszyk, C., and Urban, J. (2017). Monte carlo tableau proof search. In *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, pages 563–579.

Fuchs, M. (1995). Learning proof heuristics by adaptive parameters. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 235–243.

Fuchs, M. (1998). A feature-based learning method for theorem proving. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 457–462, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Irving, G., Szegedy, C., Alemi, A. A., Eén, N., Chollet, F., and Urban, J. (2016). DeepMath - deep sequence models for premise selection. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2235–2243.

Kaliszyk, C., Chollet, F., and Szegedy, C. (2017). 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.*

Kaliszyk, C. and Urban, J. (2013). Stronger automation for Flyspeck by feature weighting and strategy evolution. In *Third International Workshop on Proof Exchange for Theorem Proving, PxTP 2013, Lake Placid, NY, USA, June 9-10, 2013*, pages 87–95.

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

Kaliszyk, C. and Urban, J. (2015a). FEMaLeCoP: Fairly efficient machine learning connection prover. In *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings*, pages 88–96.

Kaliszyk, C. and Urban, J. (2015b). Mizar 40 for mizar 40. *J. Autom. Reasoning*, 55(3):245–256.

Kaliszyk, C., Urban, J., Michalewski, H., and Olšák, M. (2018). 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.

Kaliszyk, C., Urban, J., and Vyskocil, J. (2014). Machine learner for automated reasoning 0.4 and 0.5. In *4th Workshop on Practical Aspects of Automated Reasoning, PAAR@IJCAR 2014, Vienna, Austria, 2014*, pages 60–66.

Komendantskaya, E., Heras, J., and Grov, G. (2012). Machine learning in Proof General: Interfacing interfaces. In *Proceedings 10th International Workshop*

On User Interfaces for Theorem Provers, UITP 2012, Bremen, Germany, July 11th, 2012., pages 15–41.

Kühlwein, D., Blanchette, J. C., Kaliszyk, C., and Urban, J. (2013). MaSh: Machine Learning for Sledgehammer. In *ITP*.

Kühlwein, D., van Laarhoven, T., Tsivtsivadze, E., Urban, J., and Heskes, T. (2012). Overview and evaluation of premise selection techniques for large theory mathematics. In *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, pages 378–392.

Loos, S. M., Irving, G., Szegedy, C., and Kaliszyk, C. (2017). Deep network guided proof search. In *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, pages 85–105.

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

Rocktäschel, T. and Riedel, S. (2017). End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3788–3800.

Sutcliffe, G. and Puzis, Y. (2007). SRASS - A semantic relevance axiom selection system. In *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, July 17-20, 2007, Proceedings*, pages 295–310.

Urban, J. (2004). MPTP - motivation, implementation, first experiments. *J. Autom. Reasoning*, 33(3-4):319–339.

Urban, J. (2006). MizarMode - an integrated proof assistance tool for the mizar way of formalizing mathematics. *J. Applied Logic*, 4(4):414–427.

Urban, J. (2007). MaLARea: a metasystem for automated reasoning in large theories. In *Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories, Bremen, Germany, 17th July 2007.*

Urban, J., Sutcliffe, G., Pudlák, P., and Vyskočil, J. (2008). MaLARea SG1 - machine learner for automated reasoning with semantic guidance. In Armando, A., Baumgartner, P., and Dowek, G., editors, *Automated Reasoning*, pages 441–456, Berlin, Heidelberg. Springer Berlin Heidelberg.

Urban, J., Vyskocil, J., and Stepánek, P. (2011). MaLeCoP machine learning connection prover. In *Automated Reasoning with Analytic Tableaux and Related Methods - 20th International Conference, TABLEAUX 2011, Bern, Switzerland, July 4-8, 2011. Proceedings*, pages 263–277.

Wang, M., Tang, Y., Wang, J., and Deng, J. (2017). Premise selection for theorem proving by deep graph embedding. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 2786–2796. Curran Associates, Inc.