Layered Clause Selection for Theory Reasoning (Short Paper)

Bernhard Gleiss[1(✉)] and Martin Suda[2]

[1] TU Wien, Vienna, Austria
bgleiss@forsyte.at
[2] Czech Technical University in Prague, Prague, Czech Republic

Abstract. Explicit theory axioms are added by a saturation-based theorem prover as one of the techniques for supporting theory reasoning. While simple and effective, adding theory axioms can also pollute the search space with many irrelevant consequences. As a result, the prover often gets lost in parts of the search space where the chance to find a proof is low. In this paper, we describe a new strategy for controlling the amount of reasoning with explicit theory axioms. The strategy refines a recently proposed two-layer-queue clause selection and combines it with a heuristic measure of the amount of theory reasoning in the derivation of a clause. We implemented the new strategy in the automatic theorem prover Vampire and present an evaluation showing that our work dramatically improves the state-of-the-art clause-selection strategy in the presence of theory axioms.

1 Introduction

Thanks to recent advances, saturation-based theorem provers are increasingly used to reason about problems requiring quantified theory-reasoning [4,6]. One of the standard techniques to enable such reasoning is to automatically add first-order axiomatisations of theories detected in the input [14,18]. For example, (incomplete) axiomatisations of integer and real arithmetic or McCarthy's axioms of the theory of arrays [15] are routinely used. While this simple technique is often effective, we observed (see also [21]) two problems inherent to the solution: First, explicit axioms blow up the search space in the sense that a huge amount of consequences can additionally be generated. This happens since theory axioms are often repeatedly combined with certain clauses or among themselves, effectively creating cyclic patterns in the derivation. Most of these consequences would immediately be classified as practically useless by humans. Second, many of the resulting consequences have small weight. This has the unfortunate effect that the age-weight clause selection heuristic [16], predominantly used by saturation-based theorem provers for guiding the exploration of the search-space, often selects these theory-focused consequences. This way the prover is getting lost in parts of the search space where the chance of finding a proof is low.

© Springer Nature Switzerland AG 2020

In this paper, we propose to limit the exploration of theory-focused consequences by extending clause selection to take into account the amount of theory reasoning in the derivation of a clause. Our solution consists of two parts. First, we propose an efficiently computable feature of clauses, which we call *th-distance*, that measures the amount of theory reasoning in the derivation of a clause (Sect. 3). Second, we turn to the general problem of incorporating a feature to a clause selection strategy. There has been an ongoing interest in this problem [24,25,28]. We take inspiration from the layered clause selection approach presented in [28] and introduce the refined notion of *multi-split queues*, which present a principled solution to the incorporation problem (Sect. 2). We finally obtain a clause selection strategy for theory reasoning by instantiating multi-split queues with the feature *th-distance*. We implemented the resulting clause selection in the state-of-the-art saturation-based theorem prover Vampire [14], and evaluate its benefits on a relevant subset of the SMT-LIB benchmark (Sect. 4).

Related Work. There are different approaches to adding support for theory reasoning to saturation-based theorem provers, either by extending the prover's inference system with dedicated inference rules [2,10,12,13] or using even more fundamental design changes [1,7,20,22]. While such solutions can result in very efficient reasoning procedures, their development is incredibly challenging and their implementation is a huge effort. As a result, only a few theories are covered by such approaches, in contrast to our technique, which applies to arbitrary theories. In particular, our technique can be used by non-experts on custom theory-domains coming from applications for which no dedicated solution exists. Our work has similar motivation to [21], where the authors use the set-of-support strategy [30] to limit the amount of reasoning performed with pure theory consequences. However, unlike our technique, they do not impose any limit on clauses whose derivation contains at least one non-theory-axiom.

Contributions. The summarized contributions of this paper are:

- A new approach for building clause selection strategies from clause features, based on *multi-split queues*.
- A new clause selection strategy for theory reasoning based on the instantiation of multi-split queues with the *th-distance*-feature measuring the amount of theory reasoning in the derivation of a clause. Our solution applies to arbitrary theories and does not require fundamental changes to the implementation of clause selection.
- An implementation of the introduced clause selection strategy in the state-of-the-art theorem prover Vampire.
- An experimental evaluation confirming the effectiveness of the technique, by improving on the existing heuristics by up to 37 % on a relevant set of benchmarks.

2 Layered Clause Selection

We assume the reader to be familiar with the saturation-based theorem proving technology (see, e.g. [3,17]) and, in particular, with clause selection, the procedure for deciding, at each iteration of a saturation algorithm, which of the currently passive clauses to next select for activation, i.e. for participation in inferences with the previously activated clauses. To agree on terminology, we start this section by recalling clause selection by age and weight. We then move on to explaining layered clause selection.

The two most important features of a clause for clause selection are 1) its *age*, typically implemented using an ever-increasing "date of birth" timestamp, and 2) *weight*, which refers to the number of symbols occurring in the clause. A theorem prover prefers to select clauses that are old, which implicitly corresponds to a breadth-first search strategy, and clauses that are light, which is a form of best-first search (clauses with few symbols are cheaper to process, tend to be stronger simplifiers, and are intuitively closer to the ultimate target, the empty clause). In practice, the best performance is achieved by combining these two criteria [16,25]. This is achieved by storing the passive clauses in two *queues*, one sorted by age and the other by weight, and setting a ratio to specify how the selection alternates between picking from these two queues.

Layered Selection. In the system description of GKC [28], Tammet describes an idea of using two layers of queues to organise clause selection. The first layer relies on the just-described combination of selection by age and weight. In the second layer, clauses are split into disjoint groups using a certain property (e.g., "being derived from the *goal* or not" could define two groups), each group is represented by two sub-queues of the first layer, and the decision from which group to select the next clause is dictated by a new second-layer ratio. Although Tammet does not expand much on the insights behind using the layered approach, he reports it highly beneficial for the performance of GKC. In our understanding, the additional layer (in principle, there could be more than two) provides a clean way of incorporating into clause selection a new notion of what a preferred clause should be, without a priori disturbing the already established and tuned primary approach, such as selection by age and weight.[1]

Our preliminary experiments with the idea (instantiated with the derived-from-the-goal property) found it useful, but not as powerful as other goal-directed heuristics in VAMPIRE. In particular, finding a universally good ratio between the "good" clauses and the "bad" ones seemed hard. What we propose here instead (and what also led in our experiment to a greater performance gain) is to instead organise the clauses into groups with "good" ones and "all". Here the second group contains all the passive clauses and essentially represents a fallback to the original single-layer strategy. The advantage of this new take on layered selection is that a bad clause is only selected if 1) it is time to try a bad

[1] A known alternative [25] is to adapt the formula for computing weight to include a term for penalising bad clauses and still rely on selection by age and this new refined notion of weight. (See also the *non-goal weight coefficient* in [27].).

clause according to the second-layer ratio and 2) the best bad clause is also the current overall best according to the age-weight perspective. This makes picking a good second-layer ratio much easier. In particular, one can "smoothly" move (by changing the second-layer ratio) from a high preference for the "all" second-layer queue towards selecting more "good" clauses without necessarily having to select any "bad" ones.

Multi-split Queues. We propose multi-split queues to realize layered selection with second layer groups defined by a real-valued clause feature.

Definition 1. *Let μ be a real-valued clause evaluation feature such that preferable clauses have low value of $\mu(C)$. Let the* cutoffs $c_1, \ldots, c_k$ *be monotonically increasing real numbers with $c_k = \infty$, and let the ratio $r_1 : \ldots : r_k$ be a list of positive integer values. These together determine a layered selection scheme with k groups $\mathcal{C}_i = \{C | \mu(C) \leq c_i\}$ for $i = 1, \ldots, k$, such that we select from the i-th group with a frequency $r_i / (\Sigma_{j=1}^{k} r_j)$.*

It is easy to see that multi-split queues generalise the binary "good" vs "all" arrangement, since, thanks to monotonicity of the cutoffs, we have $\mathcal{C}_i \subseteq \mathcal{C}_{i+1}$. Moreover, since $c_k = \infty$, $\mathcal{C}_k$ will contain all the passive clauses.

3 Theory Part

In this section, we instantiate the idea of multi-split queues from Sect. 2 with a concrete clause evaluation feature, which measures the amount of theory reasoning in the derivation of a clause. We assume that the initial clauses given to the saturation algorithm, which we simply refer to as *axioms*, consists of non-theory axioms obtained by classifying the input problem and theory axioms added to facilitate theory reasoning.

We start by defining the fraction of theory reasoning in the derivation of a general clause. This relies on counting the number of theory axioms, resp. the number of all axioms, in the derivation-tree using running sums.

Definition 2. *For a theory axiom C, define both $thAx(C)$ and $allAx(C)$ as 1. For a non-theory axiom C, define $thAx(C)$ as 0 and $allAx(C)$ as 1. For a derived clause C with parent clauses $C_1, \ldots, C_n$, define $thAx(C)$ as $\sum_i thAx(C_i)$ and $allAx(C)$ as $\sum_i allAx(C_i)$. Finally, we set $frac(C) := thAx(C)/allAx(C)$.*

Assume now that for a given problem we expect (based on domain knowledge and experience) the fraction of theory reasoning in the final refutation $frac(\bot)$ to be at most $1/d$, for a positive integer d. Our clause evaluation feature *th-distance* measures how much $frac(C)$ exceeds the expected "maximally allowed" fraction $1/d$. More precisely, *th-distance* counts the number of non-theory axioms which the derivation of C would additionally need to contain to achieve a ratio $1/d$.

Definition 3. *The th-distance*: *Clauses* $\rightarrow \mathbb{N}$ *is defined as*

$$th - distance(C) := max(thAx(C) \cdot d - allAx(C), 0).$$

Our heuristic is based on the idea that a clause with small *th-distance* is more likely to contribute to the refutation than a clause with high *th-distance*. We therefore want to ensure that clause selection focuses on selecting clauses C with a low value $\mathit{th\text{-}distance}(C)$. We realize this with the multi-split queues (see Sect. 2), instantiating the clause evaluation feature μ by *th-distance*, resulting in a second layer clause selection strategy with parameters d, $c_1, \ldots, c_k$ and $r_1 : \ldots : r_k$.

4 Experiments

We implemented the heuristic described in Sect. 3 in Vampire (version 4.4). Our newly added implementation consists of about 900 lines of C++ code and is compatible with both the LRS saturation algorithm [23] and Avatar [29].

For evaluation, we used the following subset of the most recent version (as of January 2020) of SMTLIB [5]: We took all the problems from the sub-logics that contain quantification and theories, such as LIA, LRA, NRA, ALIA, UFDT, ... except for those requiring bit-vector (BV) or floating-point (FP) reasoning, currently not supported by Vampire. Subsequently, we excluded problems known to be satisfiable and those that were provable using Vampire's default strategy in 10 s either without adding theory axioms or while performing clause selection by age only. This way, we obtained 20 795 problems.[2]

Table 1. Comparing clause selection strategies on Vampire's default configuration.

Strategy	d-value	Cutoffs	Ratio	Refuted	Δbase	Δbase%
`default`	–	–	–	886	0	0.0
`layered2`	10	$23, \infty$	33:8	1112	226	25.5
`layered3`	7	$0, 30, \infty$	16:8:1	1170	284	32.1
`layered4`	8	$16, 41, 59, \infty$	84:9:2:2	1176	290	32.7

As a first experiment, we compared the number of problems solved in 10 s by the default strategy[3] and its various extensions by multi-split queues defined in Sect. 3.[4] The d-value, cutoffs and ratio values for the heuristic were selected by educated guessing and randomised hill-climbing. Table 1 lists results of the best obtained configurations. It can be seen that already with two second layer queues a substantial improvement of 25.5% over the default is achieved. Moreover, while it is increasingly more difficult to choose good values for the many parameters defining a configuration with multiple queues, their use further significantly improves the number of problems solved.

[2] A list of the selected problems along with other information needed to reproduce our experiments can be found at https://git.io/JvqhP.

[3] The default strategy uses Avatar [29], the LRS saturation algorithm [23] and an age-weight ratio of 1:1.

[4] The experiment was run on our local server with Intel Xeon 2.3 GHz processors.

Table 2. Comparing clause selection strategies on Vampire's portfolio configuration.

Strategy	d-value	Cutoffs	Ratio	Refuted	Uniques
`SMTCOMP2019`	–	–	–	5479	194
`SMTCOMP2019+layered4`	8	$16, 41, 59, \infty$	84:9:2:2	5629	344

In a second experiment,[5] we ran Vampire's strategy schedule for SMTCOMP 2019 [11] on our problems and also the same schedule additionally imposing the most successful second-layer clause selection scheme `layered4` from the first experiment. The time limit was 500 s per problem. Table 2 shows the results.

We can see that the version with second-layer queues improved over the standard schedule by 150 solved problems. This is a very significant result, suggesting the achieved control of theory reasoning is incredibly helpful. Moreover, one should keep in mind that strategies in a schedule are carefully selected to complement each other and even locally good changes in the strategies often destroy this complementarity (cf., e.g., [19,21]). In our case, however, we achieve an improvement despite this looming negative effect. Finally, it is very likely that a new schedule, constructed while taking our new technique into account, will be able to additionally cover some of the 194 problems currently only solved by the unaltered schedule.

5 Conclusion

We introduced a new clause selection heuristic for reasoning in the presence of explicit theory axioms. The heuristic is based on the combination of multi-split queues and a new clause-feature measuring the amount of theory reasoning in the derivation of a clause. Our experiments show that the new heuristic significantly improves the existing state-of-the-art clause selection strategy. As future work, we want to extend layered clause selection with new clause-features and combine it with the machine-learning-based approach in the style of ENIGMA [8].

References

1. Althaus, E., Kruglov, E., Weidenbach, C.: Superposition modulo linear arithmetic SUP(LA). In: Ghilardi, S., Sebastiani, R. (eds.) FroCoS 2009. LNCS (LNAI), vol. 5749, pp. 84–99. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04222-5_5
2. Bachmair, L., Ganzinger, H.: Ordered chaining calculi for first-order theories of transitive relations. J. ACM **45**(6), 1007–1049 (1998)
3. Bachmair, L., Ganzinger, H., McAllester, D.A., Lynch, C.: Resolution theorem proving. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 2, pp. 19–99. Elsevier and MIT Press, Cambridge (2001)

[5] The second experiment was run on the StarExec cluster [26] with 2.4 GHz processors.

4. Backes, J., et al.: Reachability analysis for AWS-based networks. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11562, pp. 231–241. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25543-5_14
5. Barrett, C., Fontaine, P., Tinelli, C.: The Satisfiability Modulo Theories Library (SMT-LIB) (2016). www.SMT-LIB.org
6. Barthe, G., Eilers, R., Georgiou, P., Gleiss, B., Kovcs, L., Maffei, M.: Verifying relational properties using trace logic. In: 2019 Formal Methods in Computer Aided Design (FMCAD), pp. 170–178, October 2019
7. Baumgartner, P., Waldmann, U.: Hierarchic superposition with weak abstraction. In: Bonacina, M.P. (ed.) CADE 2013. LNCS (LNAI), vol. 7898, pp. 39–57. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38574-2_3
8. Chvalovský, K., Jakubuv, J., Suda, M., Urban, J.: ENIGMA-NG: efficient neural and gradient-boosted inference guidance for E. In: Fontaine [9], pp. 197–215
9. Fontaine, P. (ed.): CADE 2019. LNCS (LNAI), vol. 11716. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6
10. Gupta, A., Kovács, L., Kragl, B., Voronkov, A.: Extensional crisis and proving identity. In: Cassez, F., Raskin, J.-F. (eds.) ATVA 2014. LNCS, vol. 8837, pp. 185–200. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11936-6_14
11. Hadarean, L., Hyvarinen, A., Niemetz, A., Reger, G.: 14th International Satisfiability Modulo Theories Competition (SMT-COMP 2019) (2019). https://smt-comp.github.io/2019/
12. Kotelnikov, E., Kovács, L., Reger, G., Voronkov, A.: The vampire and the FOOL. In: Avigad, J., Chlipala, A. (eds.) Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs, Saint Petersburg, FL, USA, 20–22 January 2016, pp. 37–48. ACM (2016)
13. Kovács, L., Robillard, S., Voronkov, A.: Coming to terms with quantified reasoning. In: Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages. POPL 2017, New York, NY, USA, pp. 260–270. Association for Computing Machinery (2017)
14. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_1
15. Mccarthy, J.: Towards a mathematical science of computation. In: IFIP Congress, pp. 21–28. North-Holland (1962)
16. McCune, W.: Otter 3.0 reference manual and guide. Technical report ANL-94/6, Argonne National Laboratory (1994)
17. Overbeek, R.A.: A new class of automated theorem-proving algorithms. J. ACM **21**(2), 191–200 (1974)
18. Prevosto, V., Waldmann, U.: SPASS+T. In: Sutcliffe, G., Schmidt, R., Schulz, S. (eds.) Proceedings of the FLoC 2006 Workshop on Empirically Successful Computerized Reasoning, 3rd International Joint Conference on Automated Reasoning, number 192 in CEUR Workshop Proceedings, pp. 19–33 (2006)
19. Rawson, M., Reger, G.: Old or heavy? Decaying gracefully with age/weight shapes. In: Fontaine [9], pp. 462–476
20. Reger, G., Bjorner, N., Suda, M., Voronkov, A.: AVATAR modulo theories. In: Benzmüller, C., Sutcliffe, G., Rojas, R. (eds.) GCAI 2016. 2nd Global Conference on Artificial Intelligence, EPiC Series in Computing, vol. 41, pp. 39–52. EasyChair (2016)

21. Reger, G., Suda, M.: Set of support for theory reasoning. In: Eiter, T., Sands, D., Sutcliffe, G., Voronkov, A. (eds.) IWIL@LPAR 2017 Workshop and LPAR-21 Short Presentations, Maun, Botswana, 7–12 May 2017, vol. 1. Kalpa Publications in Computing. EasyChair (2017)
22. Reger, G., Suda, M., Voronkov, A.: Unification with abstraction and theory instantiation in saturation-based reasoning. In: Beyer, D., Huisman, M. (eds.) TACAS 2018. LNCS, vol. 10805, pp. 3–22. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89960-2_1
23. Riazanov, A., Voronkov, A.: Limited resource strategy in resolution theorem proving. J. Symb. Comput. **36**(1–2), 101–115 (2003)
24. Schulz, S., Cruanes, S., Vukmirovic, P.: Faster, higher, stronger: E 2.3. In Fontaine [9], pp. 495–507
25. Schulz, S., Möhrmann, M.: Performance of clause selection heuristics for saturation-based theorem proving. In: Olivetti, N., Tiwari, A. (eds.) IJCAR 2016. LNCS (LNAI), vol. 9706, pp. 330–345. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40229-1_23
26. Stump, A., Sutcliffe, G., Tinelli, C.: StarExec, a cross community logic solving service (2012). https://www.starexec.org
27. Suda, M.: Aiming for the goal with SInE. In: Kovacs, L., Voronkov, A. (eds.) Vampire 2018 and Vampire 2019. The 5th and 6th Vampire Workshops. EPiC Series in Computing, vol. 71, pp. 38–44. EasyChair (2020)
28. Tammet, T.: GKC: a reasoning system for large knowledge bases. In: Fontaine [9], pp. 538–549
29. Voronkov, A.: AVATAR: the architecture for first-order theorem provers. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 696–710. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_46
30. Wos, L., Robinson, G.A., Carson, D.F.: Efficiency and completeness of the set of support strategy in theorem proving. J. ACM **12**(4), 536–541 (1965)