Efficient Model Construction for Horn Logic with VLog: Extended Abstract[⋆]

Jacopo Urbani[1], Markus Krötzsch[2], Ceriel Jacobs[1], Irina Dragoste[2], and David Carral[2]

[1] Vrije Universiteit Amsterdam, The Netherlands, firstname@cs.vu.nl
[2] cfaed, TU Dresden, Germany, firstname.lastname@tu-dresden.de

Horn ontologies consisting of *existential rules* are used in various fields ranging from reasoning over knowledge graphs [7] and Description Logics (DL) ontologies [5,6], to data integration [4] and social network analysis [10]. To solve conjunctive query answering over these logical theories, we can apply the *chase algorithm*—a sound and complete (albeit non-terminating) bottom-up materialisation procedure where all relevant consequences are precomputed, allowing queries to be directly evaluated over materialised sets of facts.

As our main contribution, we extend the in-memory Datalog engine VLog [11] to support Horn existential rules without equality (a fragment that encompasses *Horn-*$\mathcal{SRI}$ in terms of expressivity). Namely, we implement the *skolem* and the *restricted variants of the chase* on VLog's architecture. In the *skolem chase*, rules are replaced by their skolemisation. In the *restricted chase*, new terms are introduced during the reasoning process only if already derived terms and facts cannot be reused to satisfy the corresponding existential restriction. The latter terminates in many more cases than the former [2,3] and often produces smaller models, but termination depends on the rule application order and its implementation requires value reusability checks. We implement a slightly different version of the restricted chase which leads to termination in more cases [3], by prioritising the exhaustive application of Datalog rules (rules without existentially quantified variables). This enables facts derived from Datalog rules to satisfy some existential restrictions that would otherwise lead to non-termination.

In our implementation, we exploit the highly memory-efficient architecture of VLog, based on columnar storage: instead of storing a list of tuples (rows), the data is organised into a tuple of columns (value lists). The columns are ordered lexicographically, enabling fast merge joins and duplicate elimination, as well as data compression schemes for low memory usage. Because updates are slow in columnar tables, VLog operates in append-only mode, applying one rule per materialisation step, and creating separate tables for the derived facts. To reduce redundant derivations, VLog uses *semi-naive evaluation*, which only considers rule body matches that were not found up to the previous application of the same rule. We adopted the *1-parallel-restricted chase* [1] optimisation, in which the facts derived in the ongoing chase step are not checked for value reusability.

We evaluate our implementation using existential rule programs from a recent (skolem and restricted) chase benchmark [1]. In addition, we also use rules obtained from translating data-rich, real world OWL ontologies (UOBM, Reactome, and Uniprot). The test data involves programs with millions of facts and thousands of rules, and predicates with relatively large arities (maximum 11). We test increasing partitions of data for the

[⋆] The full version of this paper was published at IJCAR 2018 [12].

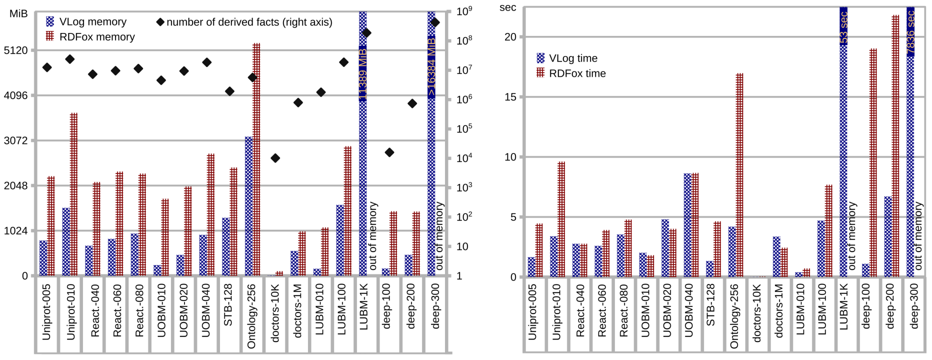

Fig. 1. Memory usage (left) and materialisation time (right) for VLog and RDFox for the restricted chase

same rule sets. In each case, we run the skolem and restricted chase on a commodity laptop, and measure time and peak memory usage. We compare the results against the chase benchmark leading system, RDFox [9,8]. Like VLog, RDFox is a performance-oriented in-memory chase engine, but it is designed with the goal of enabling highly parallel processing rather than memory efficiency. RDFox does not prioritise Datalog rules, hence may terminate in fewer cases than VLog, but this does not affect our experiments.

The results for the restricted chase are shown in Figure 1. Both VLog and RDFox require significantly less time and memory for the restricted chase, with only few exceptions (*deep-100*, *deep-200*, *Ontology-256* require slightly less time and memory in the skolem chase). These findings further motivate the use of the restricted chase.

On average, VLog uses only 40% of the memory required by RDFox, ostensibly due to its compressed data structures. In two cases, RDFox runs out of memory, while VLog terminates (using OS swap space in only one case). Regarding time performance, VLog takes between 5.8% and 137.5% of the time needed by RDFox. RDFox outperforms VLog for only two ontologies, *UOBM* and *doctors*. This result is surprising, since VLog ran on a single thread, while RDFox used maximal parallelism (8 cores), in some cases peaking at over 700% of CPU usage.

To summarise, we provide the first implementation of the chase on a columnar architecture, by extending VLog. Our tool is more memory-efficient than the state of the art while being comparatively fast, even on a single thread. VLog is free and open-source. It can be used in two ways: as a command-line client with an optional web interface,[3] and through a Java API[4] with additional functionality (translating OWL files to rules, executing SPARQL queries against remote endpoints, integrating RDF data, . . .). We hope our tool proves useful for the community, and encourages a broader adoption of the expressive language of existential rules.

Acknowledgements This work was supported by the DFG within the cfaed Cluster of Excellence, CRC 912 (HAEC), and Emmy Noether grant KR 4381/1-1.

[3] VLog core reasoner: `https://github.com/karmaresearch/vlog`
[4] VLog4j: `https://github.com/knowsys/vlog4j`, also available via Maven

References

1. Benedikt, M., Konstantinidis, G., Mecca, G., Motik, B., Papotti, P., Santoro, D., Tsamoura, E.: Benchmarking the chase. In: Proc. 36th Symposium on Principles of Database Systems (PODS'17). pp. 37–52. ACM (2017)
2. Carral, D., Dragoste, I., Krötzsch, M.: Restricted chase (non)termination for existential rules with disjunctions. In: Sierra, C. (ed.) Proc. 26th Int. Joint Conf. on Artificial Intelligence (IJCAI'17). pp. 922–928. IJCAI (2017)
3. Carral, D., Feier, C., Hitzler, P.: A practical acyclicity notion for query answering over Horn-$\mathcal{SRIQ}$ ontologies. In: Proc. 15th Int. Semantic Web Conf. (ISWC'16). LNCS, vol. 9981, pp. 70–85 (2016)
4. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theoretical Computer Science 336(1), 89–124 (2005)
5. Kazakov, Y.: Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In: Proc. 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09). pp. 2040–2045. IJCAI (2009)
6. Kazakov, Y., Krötzsch, M., Simančík, F.: The incredible ELK: From polynomial procedures to efficient reasoning with $\mathcal{EL}$ ontologies. J. of Automated Reasoning 53, 1–61 (2013)
7. Marx, M., Krötzsch, M., Thost, V.: Logic on MARS: Ontologies for generalised property graphs. In: Sierra, C. (ed.) Proc. 26th Int. Joint Conf. on Artificial Intelligence (IJCAI'17). pp. 1188–1194. IJCAI (2017)
8. Motik, B., Nenov, Y., Piro, R., Horrocks, I.: Combining rewriting and incremental materialisation maintenance for datalog programs with equality. In: Yang, Q., Wooldridge, M. (eds.) Proc. 24th Int. Joint Conf. on Artificial Intelligence (IJCAI'15). pp. 3127–3133. AAAI Press (2015)
9. Motik, B., Nenov, Y., Piro, R., Horrocks, I., Olteanu, D.: Parallel materialisation of Datalog programs in centralised, main-memory RDF systems. In: Proc. 28th AAAI Conf. on Artif. Intell. (AAAI'14). pp. 129–137. AAAI Press (2014)
10. Seo, J., Guo, S., Lam, M.S.: SociaLite: an efficient graph query language based on Datalog. IEEE Trans. Knowl. Data Eng. 27(7), 1824–1837 (2015)
11. Urbani, J., Jacobs, C., Krötzsch, M.: Column-oriented Datalog materialization for large knowledge graphs. In: Proc. 30th AAAI Conf. on Artificial Intelligence (AAAI'16). pp. 258–264. AAAI Press (2016)
12. Urbani, J., Krötzsch, M., Jacobs, C.J.H., Dragoste, I., Carral, D.: Efficient model construction for horn logic with vlog - system description. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) Proc. 9th Int. Joint Conf. on Automated Reasoning (IJCAR'18). LNCS, vol. 10900, pp. 680–688. Springer (2018)