dlv – An Overview *

Robert Bihlmeyer Wolfgang Faber Christoph Koch
Nicola Leone Cristinel Mateis Gerald Pfeifer†

Institut für Informationssysteme, TU Wien
Paniglgasse 16, A-1040 Wien, Austria
{*robbe,faber,koch,leone,mateis,pfeifer*}*@dbai.tuwien.ac.at*

dlv [2, 3] is a knowledge representation system, based on disjunctive logic programming, which offers front-ends to several advanced KR formalisms. The system has been developed since the end of 1996 at Technische Universität Wien in an ongoing project funded by the Austrian Science Funds.

dlv is based on very solid theoretical foundations [5], and recent comparisons [3] have shown that dlv is nowadays a state-of-the-art implementation of disjunctive logic programming. A major strength of dlv are its advanced knowledge modelling features. Its kernel language, which extends disjunctive logic programming by strong negation (à la Gelfond and Lifschitz) and integrity constraints, allows for representing complex knowledge based problems in an elegant and highly declarative fashion [3].

The computational kernel of dlv is an efficient engine for computing the stable models of a disjunctive logic program, which efficiently implements the algorithms presented in [5], improving them by several optimization techniques. The system runs in polynomial space and single exponential time in the worst case, and is able to efficiently recognize and process syntactical subclasses of disjunctive logic programs which have lower computational complexity than the general case (like, e.g., programs with head-cycle free disjunction or stratified negation). Enhanced modular evaluation techniques are currently under development along with linguistic extensions to deal also with quantitative information [1, 4].

The three main modules of the dlv kernel are the Intelligent Grounding, the Model Generator and the Model Checker. All of these modules perform a modular evaluation of their input according to various dependency graphs as defined in [5, 2] and try to detect and efficiently handle special (syntactic) subclasses, which in general yields a tremendous speedup.

*Supported by *FWF (Austrian Science Funds)* under the project P11580-MAT "A Query System for Disjunctive Deductive Databases"

†Please address correspondence to this author.

The Intelligent Grounding takes an input program, whose facts can be stored also in the tables of external relational databases, and efficiently generates a subset of the program instantiation that has exactly the same stable models as the full program, but is much smaller in general. (For stratified programs, for example, the Grounding already computes the single stable model.)

Then the Model Generator is run on the (ground) output of the Intelligent Grounding. It generates one candidate for a stable model at a time and invokes the Model Checker to verify whether it is indeed a stable model.

The Model Generator computes first the least fixpoint $W_P^\infty(\emptyset)$ of a deterministic operator W_P [5], which extends the well-founded operator of Van Gelder, Ross and Schlipf to the disjunctive case. If $W_P^\infty(\emptyset)$ is a total interpretation, then it is the unique stable model [5], which can be returned to the user without any further check.

Otherwise, $W_P^\infty(\emptyset)$ is contained in every stable model [5] and is used as the deterministic basis for a backtracking algorithm that makes nondeterministic choices followed by the computation of all direct consequences from these choices.

Candidate stable models produced by the Model Generator are checked for stability by the Model Checker of `dlv`. Verifying the stability condition is a difficult task in general, as it is a well-known co-NP-hard problem (i.e., basically, as hard as inference from the stable models of a non-disjunctive program). Instead of rewriting the program by the Gelfond-Lifschitz transformation and then checking the minimality of the candidate stable model on the rewritten program, the Model Checker verifies the stability directly, by checking whether the candidate stable model is unfounded-free ([5]).

References

[1] F. Buccafurri, N. Leone, and P. Rullo. Strong and Weak Constraints in Disjunctive Datalog. In *Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR '97)*, Dagstuhl, Germany, July 1997.

[2] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A Deductive System for Nonmonotonic Reasoning. In J. Dix, U. Furbach, and A. Nerode, editors, *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR '97)*, number 1265 in Lecture Notes in AI (LNAI), Berlin, 1997. Springer.

[3] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. The KR System `dlv`: Progress Report, Comparisons and Benchmarks. In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *Proceedings Sixth In-*

ternational Conference on Principles of Knowledge Representation and Reasoning (KR'98), pages 406–417. Morgan Kaufmann Publishers, 1998.

[4] W. Faber. Disjunctive datalog with strong and weak constraints: Representational and computational issues. Master's thesis, Institut für Informationssysteme, Technische Universität Wien, 1998.

[5] N. Leone, P. Rullo, and F. Scarcello. Disjunctive stable models: Unfounded sets, fixpoint semantics and computation. *Information and Computation*, 135(2):69–112, June 1997.