Classes of Logic Programs which Possess Unique Supported Models

Pascal Hitzler[1] and **Anthony Karel Seda**
Department of Mathematics
University College, Cork, Ireland
{phitzler,aks}@ucc.ie, http://maths.ucc.ie/~{pascal,seda}/

Logic programming is concerned with the use of logic as a programming language. The main manifestation of this computing paradigm is in the various versions of Prolog which are now available, in which computation is viewed as deduction from sets of Horn clauses, although there is also growing interest in the related form known as answer set programming, see [10]. The reference [2] contains a good survey of the growth of logic programming over the last twenty five years both as a stand-alone programming language and as a software component of large information systems.

One advantage a logic program P has over conventional imperative and object oriented programs is that it has a natural machine-independent meaning, namely, its logical meaning. This is often referred to as its declarative semantics, and is usually taken to be some "standard" model canonically associated with P. Unfortunately, it is often the case that there are many possible choices for the standard model, some even taken in many-valued logic, which do not in general coincide and all of which have a claim to be "the natural choice" depending on one's view of non-monotonic reasoning [6, 7, 11]. However, most two-valued semantics which are proposed are refinements of a particular model called the *supported model semantics* or *Clark completion semantics* [5], which we study in this paper. Indeed, we focus on programs which have a unique reading under this semantics or, in other words, programs which have a unique supported model and which are called *uniquely determined* programs. Several classes of programs with this property have been studied in the literature including acceptable programs [1], which are of great importance in connection with termination, and the acyclic and the locally hierarchical programs [4] studied from the point of view of denotational semantics. In general, such classes as these are defined by restricting their syntax using well-founded orders.

Starting from these results, our discussion will revolve around the following aspects of classes of programs, which we call *unique supported model classes* or *USM classes*, each member of which has a unique supported model.

(1) By means of operators in three-valued logic, it is possible to obtain a unifying framework for the USM classes mentioned above, see [8]. Using this observation, results which apply to these classes separately can be generalized to a considerable extent. This approach gives rise to the USM class of Φ^*-*accessible* programs. This class is remarkable in that not only does each program in it have a unique supported model, but in addition it is computationally adequate in the sense that every partial recursive function can be implemented, under Prolog, by such programs. This does not hold, for example, for the subclass of acceptable programs which always terminate under the Prolog selection rule.

(2) The general classes arrived at in (1) are susceptible to a semantical analysis using fixed-point theorems from general topology. The application of these theorems yields strong results on how to obtain the supported-model semantics by a process involving

[1]The first named author acknowledges financial support under grant SC/98/621 from Enterprise Ireland.

limits in spaces which carry a notion of distance. Results of this kind make it possible to study logic programs in the context of neural networks [9] or chaotic dynamical systems [12].

(3) It is reasonable to expect that many of the standard semantics will coincide for uniquely determined programs, so that they can be viewed as semantically unambiguous. A formal analysis shows that this is the case for the Φ^*-accessible programs, for example, and our results provide some interconnections between the various semantics which have been proposed, although this issue has yet to be fully understood and is much investigated in the current literature.

References

[1] Apt, K.R. and Pedreschi, D. Reasoning about Termination of Pure Prolog Programs. Information and Computation **106** (1993), 109–157.

[2] Apt, K.R., Marek, V.W., Truszczynski, M. and Warren, D.S. The Logic Programming Paradigm: A 25-Year Perspective. Springer, Berlin, 1999.

[3] Batarekh, A. and Subrahmanian, V.S. Topological Model Set Deformations in Logic Programming. Fundamenta Informaticae **12** (1989), 357–400.

[4] Cavedon, L. Continuity, Consistency, and Completeness Properties for Logic Programs. In: Levi, G. and Martelli, M. (Eds.), Proc. 6th International Conference on Logic Programming. MIT Press, Cambridge MA (1989), pp. 571–584.

[5] Clark, K.L. Negation as Failure. In: Gallaire, H. and Minker, J. (Eds.), Logic and Data Bases. Plenum Press, New York (1978), pp. 293–322.

[6] Van Gelder, A., Ross, K.A. and Schlipf, J.S. The Well-Founded Semantics for General Logic Programs. Journal of the ACM **38** (3) (1991), 620–650.

[7] Gelfond, G. and Lifschitz, V. The Stable Model Semantics for Logic Programming. In: Kowalski, R.A. and Bowen, K.A. (Eds.), Logic Programming. Proceedings of the 5th International Conference and Symposium on Logic Programming. MIT Press (1988), pp. 1070–1080.

[8] Hitzler, P. and Seda, A.K. Characterizations of Classes of Programs by Three-Valued Operators. In: Gelfond, M., Leone, N. and Pfeifer, G. (Eds.), Logic Programming and Nonmonotonic Reasoning, Proceedings of the 5th International Conference on Logic Programming and Non-Monotonic Reasoning, El Paso, Texas, USA, December 1999. Lecture Notes in Artificial Intelligence, Vol. 1730. Springer, Berlin (1999), pp. 357–371.

[9] Hölldobler, S., Störr, H. and Kalinke, Y. Approximating the Semantics of Logic Programs by Recurrent Neural Networks, Applied Intelligence **11** (1999), 45–58.

[10] Marek, V.M. and Truszczyński, M. Stable Models and an Alternative Logic Programming Paradigm. In: Apt, K.R., Marek, V.W., Truszczyński, M., Warren, D.S. (Eds.), The Logic Programming Paradigm. Springer, Berlin (1999), pp. 375–398.

[11] Przymusinska, H. and Przymusinski, T.C. Weakly Stratified Logic Programs. In: Apt, K.R. (Ed.), Special issue of Fundamenta Informaticae on Logical Foundations of Artificial Intelligence **13** (1990), 51–65.

[12] Seda, A.K. and Hitzler, P. Strictly Level-decreasing Logic Programs. In: Butterfield, A. and Flynn, S. (Eds.), Proceedings of the 2nd Irish Workshop on Formal Methods, Cork, 1998. Electronic Workshop in Computing, British Computer Society, 1999, pp. 1–18.