Submitted to the Technical Communications of the International Conference on Logic Programming (ICLP'10)
http://www.floc-conference.org/ICLP-home.html

METHODS AND METHODOLOGIES FOR DEVELOPING ANSWER-SET PROGRAMS—PROJECT DESCRIPTION

JOHANNES OETSCH, JÖRG PÜHRER, AND HANS TOMPITS

Technische Universität Wien,
Institut für Informationssysteme 184/3,
Favoritenstraße 9–11, A–1040 Vienna, Austria
E-mail address: {oetsch,puehrer,tompits}@kr.tuwien.ac.at

Abstract. Answer-set programming (ASP) is a well-known formalism for declarative problem solving, enjoying a continuously increasing number of diverse applications. However, arguably one of the main challenges for a wider acceptance of ASP is the need of tools, methods, and methodologies that support the actual programming process. In this paper, we review the main goals of a project, funded by the Austrian Science Fund (FWF), which aims to address this aspect in a systematic manner. The project is planned for a duration of three years and started in September 2009. Generally, the focus of research will be on methodologies for *systematic program development*, *program testing*, and *debugging*. In particular, in working on these areas, special emphasis shall be given to the ability of the developed techniques to respect the declarative nature of ASP. To support a sufficient level of usability, solutions are planned to be compatible not only for the core language of ASP but also for important extensions thereof that are commonly used and realised in various answer-set solvers. Ultimately, the methods resulting from the project shall form the basis of an *integrated development environment* (IDE) for ASP that is envisaged to combine straightforward as well as advanced techniques, realising a convenient tool for developing answer-set programs.

1. Introduction

Answer-set programming (ASP), in its most important incarnation as *logic programming under the answer-set semantics* [Gel88, Gel91a], is a well-known paradigm for declarative problem solving whose underlying idea is that problems are solved by encoding them in terms of programs such that the solutions of a given problem are determined by the models of the associated program. Hence, the models of the latter provide the "answers" of the input problems—this stands in contrast to traditional logic-based knowledge representation where *proofs* constitute an answer to a problem. The development of sophisticated answer-set solvers (see, e.g., Denecker *et al.* [Den09] for a recent overview), led to successful applications in diverse fields like planning [Gel02], diagnosis [Eit99, Gel01], symbolic model checking [Hel03], bioinformatics [Pal09, Erd09b, Erd09a], e-tourism [Iel09], patient monitoring [Mil09], music composition [Boe08, Boe09], and many others.

Key words and phrases: answer-set programming, program development, testing, debugging.
This work was partially supported by the Austrian Science Fund (FWF) under grant P21698.

Although ASP is regarded as a programming paradigm, it currently offers only limited support for *developing programs*, compared to other programming languages for which a large number of tools and development methodologies exists. Indeed, ASP research so far concentrated, by and large, on (i) formal properties of the answer-set semantics, (ii) issues related to using it for knowledge representation and reasoning, and (iii) the development of ASP solvers. These endeavours led to the acceptance of ASP as a viable approach for declarative knowledge representation, but in order to achieve a wider acceptance of ASP among practising software and knowledge engineers, *tools and methods for supporting the programmer in developing programs are needed*. Although already more than a decade ago, De Schreye and Denecker [DS99] identified this need as being vital towards practicability of computational logic formalisms in general, only recently increased efforts in this direction have started in the ASP community—particularly on debugging and modularity aspects [Bra05, Pon09, Syr06, Bra07, Mik07, Geb08]. The awareness of these engineering-oriented requirements is also reflected by the launch of the SEA workshop series on *Software Engineering for Answer-Set Programming* in 2007.[1]

In this paper, we review the main goals of the project "Methods and Methodologies for Developing Answer-Set Programs" which we started in September 2009. It aims to address the above mentioned engineering requirements for ASP in a systematic manner and is hosted by the Knowledge-Based Systems Group of the Institute of Information Systems at the Vienna University of Technology. Funding is provided by the Austrian Science Fund (FWF) and the planned duration of the project is three years. It comprises two researcher positions and is lead by Hans Tompits.

Generally, the focus of research of the project will be on methodologies for *systematic program development*, *program testing*, and *debugging*. We want to study *abstract concepts* that underlie the tasks emerging when programming in ASP as well as to *develop concrete tools* realising the researched techniques. In particular, special emphasis shall be given to the ability of the developed techniques to respect the declarative nature of ASP. Furthermore, the theoretical line of research will also involve aspects of decidability and complexity, and, concerning the development of tools, we plan to incorporate proof-of-concept implementations of the theoretical formalisms and approaches that will emerge from our research into an *integrated development environment* (IDE). The evolution of this system shall be subject to continuous evaluation in the context of laboratory courses on logic programming (such courses are held each semester at our institute).

We also plan research cooperations with different ASP groups, in particular with those at the University of Potsdam (Germany), Aalto University (Finland; incorporating the former Helsinki University of Technology), the University of Bath (U.K.), and the University of Calabria (Italy).

The next section provides a more detailed discussion of our research agenda.

2. Project objectives

2.1. Systematic program development

We want to investigate methods which support the systematic development of programs. In standard programming languages, *incremental* and *iterative program development* are

[1] See `sea07.cs.bath.ac.uk` and `sea09.cs.bath.ac.uk`.

well-known techniques. In the first method, a program is divided into subparts by functionality and developed one by one, whilst in the second method, a complex program is developed by iteratively refining the functionalities of less complex versions of it. We plan to study incremental and iterative development methods in the context of ASP. Here, the notion of a *conservative extension* [Gel91b, Gel96], which was introduced as a theoretical basis for the enhancement of a logic program by adding further rules, can be abstracted by considering program extensions in terms of *operators* prescribing how a program can be enhanced along with a binary *correspondence relation* between a program and its enhancement. The role of such a correspondence relation is to reflect the property to be preserved when enhancing the original program. The issue of program correspondence has been the subject of extensive research in the recent past [Lif01, Ino04, Eit05, Oik06, Oet07, Eit07, Fin08, Oet08, Tru09] and a project, also funded by the FWF, was conducted at our research group on this topic and successfully finished in 2008.[2]

A crucial aspect for the issue of incremental program development, where larger programs are composed from smaller program parts, is the question of *program modularity* (cf., e.g., the work of Brogi *et al.* [Bro94] about logic programs without negation). We want to analyse how different notions of a program module can be used to compose complex programs or how such notions can be extended or refined. An interesting concept in this regard is the notion of a *DLP-function* [Jan09], having its roots in early work on *lp-functions* [Gel96], which, roughly speaking, is a logic program together with an interface specifying the input, output, and local atoms of that program. DLP-functions are assigned with an extension of the answer-set semantics admitting a compositional semantics, i.e., the semantics of a modularised program is given as the union (suitably defined) of the semantics of its modules. For the software support methods that will be developed in this project, we will investigate how they can be refined to module-based versions.

We also want to develop techniques that support developers during the *intermediate coding process*. In particular, we are interested in methods that provide the user with *suggestions on how to proceed* when writing a program, which are computed from the current state of the program and the desired behaviour that is captured by the specification of a program. A potential instance of a tool providing suggestions is an intelligent *code-completion technique* that offers a selection of potential endings for a rule that is currently edited. From a methodological point of view, suggestion techniques seem especially valuable in combination with a *test-driven development approach* [Bec02]. We want to analyse the suitability of development approaches where test cases are formulated in advance and then used to directly support the coding process.

2.2. Testing methodologies

Though crucial for the quality of conventional software, there is little work on *testing methodologies* for logic programs. We want to address this issue by analysing how prominent methods from software testing [Mye79, Het91], like *black-box testing* or *white-box testing*, can be adapted to ASP. We recall that the idea of black-box testing is to derive test cases from the specification of a component but to abstract from the internal structure of the component—indeed, as the name suggests, it is only used as a "black box". Important in this context is how test cases can be derived from specifications and how to rate the *quality* of a suite of test cases with respect to their ability to detect unimplemented or

[2]See `http://www.kr.tuwien.ac.at/research/projects/eq/` for details about this project.

faultily implemented parts of a specification. Complementary to black-box testing, white-box testing is based on the *structure* of a component: For conventional languages, white-box testing aims at deriving test cases that cover possible paths through a software component. As one of the first results within our project, we developed, jointly with Ilkka Niemelä and Tomi Janhunen from Aalto University, different coverage metrics for ASP, along with their mutual relationships, and laid down basic techniques for test automation using ASP itself [Nie10].

2.3. Debugging

Complementing the development and testing aspects, we want to study methods to *debug answer-set programs*, i.e., techniques supporting the programmer in *localising and fixing program errors*. Initial research in this direction is already available [Bra05, Pon09, Syr06, Bra07, Geb08] but further investigation is needed.[3] Especially, as we want to develop debugging techniques for ASP that are applicable in real-world scenarios, debugging methodologies for non-ground programs are needed—most debugging methods studied so far concentrate on propositional programs only, however. To address this issue, we extended the meta-programming technique of Gebser *et al.* [Geb08] to the non-ground case [Oet10]. Moreover, we intend to consider not only the core language of answer-set programs but also cover important language extensions like *weak constraints*, *aggregates*, or *choice rules*.

Analogous to techniques for providing suggestions during the immediate coding process, we want to investigate similar features for program debugging as well. Advice on how to fix a program can be of different nature, e.g., proposals to remove specific constraints that eliminate desired answer sets.

Another important issue we want to address is *local debugging* in connection with modular-programming concepts. We want to clarify how the search for errors can be restricted to suspicious program components and how program parts can be individually debugged, yielding correctness of the overall program.

2.4. Specifications for answer-set programs

Most of the methods we are aiming at require information about the *intended behaviour* of a program under consideration. For instance, for localising a bug in an erroneous program, a debugging system needs to be aware of what the correct semantics of the program is, in order to classify wrong behaviour. Thus, *respective methods for specifying program properties are needed*. This point is important despite the widely held view that, because of the declarative nature of ASP, logic programs can be seen as specifications themselves, which, in turn, would eliminate the ubiquitous gap between specification and programming, as argued by Baral [Bar03]. However, since the process of developing answer-set programs is not as straightforward as the latter point of view might suggest, it may prove helpful to describe the desired behaviour of a program in a way that is easier to achieve than the program itself. This may be done, e.g., in the form of sample test cases, or formally defined program properties, that only describe *certain aspects of the intended semantics*, vis-a-vis a full specification as represented by a complete program. In any case, there is certainly some

[3] It is also worthwhile to mention here the work of Wittocx *et al.* [Wit09] on debugging ID-logic theories, i.e., first-order theories with inductive definitions.

subtlety in using declarative descriptions for a declarative language which requests paying close attention to the practicability of studied specification formalisms.

2.5. Implementation

Complementing our theoretical investigations, we also plan to develop prototype implementations of our techniques. Towards providing effective support for developers, we want to incorporate these into an *integrated development environment* (IDE). A few systems for developing answer-set programs have been introduced so far [Per07, Sur07], but, despite providing useful utilities, they are still in an early state of development and leave much room for improvement. Besides the envisaged functionalities for specification, testing, and debugging, our IDE should also allow customary functions like providing a code editor, syntax checking, and syntax highlighting, which are rather trivial from a theoretical point of view but handy in use. Moreover, we want to guarantee interoperability with most of the available popular solvers. The IDE will be subject to continuous evaluation within the laboratory courses on logic programming we teach regularly at our university, and it will be made publicly available for, e.g., other universities and their teaching needs.

3. Conclusion

Providing intelligent development methodologies and tools constitutes a natural next step in the evolution of ASP and will hopefully have a positive impact on this field as a whole. Furthermore, with such techniques at hand, both expert as well as novice programmers will have an enhanced access to powerful declarative problem-solving machineries.

For further information about the project, see

`http://www.kr.tuwien.ac.at/research/projects/mmdasp/`.

References

[Bar03] Chitta Baral. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press, Cambridge, England, UK, 2003.

[Bec02] Kent Beck. *Test Driven Development: By Example*. Addison-Wesley Professional, Boston, MA, USA, 2002.

[Boe08] Georg Boenn, Martin Brain, Marina De Vos, and John Fitch. Automatic composition of melodic and harmonic music by answer set programming. In Maria Garcia de la Banda and Enrico Pontelli (eds.), *Proceedings of the 24th International Conference on Logic Programming* (*ICLP'08*), *Udine, Italy, December 9-13, 2008*, *Lecture Notes in Computer Science*, vol. 5366, pp. 160–174. Springer, Berlin-Heidelberg, Germany, 2008.

[Boe09] Georg Boenn, Martin Brain, Marina De Vos, and John Fitch. ANTON: Composing logic and logic composing. In Esra Erdem, Fangzhen Lin, and Torsten Schaub (eds.), *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'09*), *Potsdam, Germany, September 14-18, 2009*, *Lecture Notes in Computer Science*, vol. 5753, pp. 542–547. Springer, Berlin-Heidelberg, Germany, 2009.

[Bra05] Martin Brain and Marina De Vos. Debugging logic programs under the answer-set semantics. In *Proceedings of the 3rd Workshop on Answer Set Programming: Advances in Theory and Implementation* (*ASP'05*), *Bath, UK, July 27-29, 2005*, *CEUR Workshop Proceedings*, vol. 142. CEUR-WS.org, Aachen, Germany, 2005.

[Bra07] Martin Brain, Martin Gebser, Jörg Pührer, Torsten Schaub, Hans Tompits, and Stefan Woltran. Debugging ASP programs by means of ASP. In Chitta Baral, Gerhard Brewka, and John S. Schlipf (eds.), *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'07*), *Tempe, AZ, USA, May 15-17, 2007*, *Lecture Notes in Computer Science*, vol. 4483, pp. 31–43. Springer, Berlin-Heidelberg, Germany, 2007.

[Bro94] Antonio Brogi, Paolo Mancarella, Dino Pedreschi, and Franco Turini. Modular logic programming. *ACM Transactions on Programming Languages and Systems*, 16(4):1361–1398, 1994.

[Den09] Marc Denecker, Joost Vennekens, Stephen Bond, Martin Gebser, and Miroslaw Truszczynski. The second answer set programming competition. In Esra Erdem, Fangzhen Lin, and Torsten Schaub (eds.), *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'09*), *Potsdam, Germany, September 14-18, 2009*, *Lecture Notes in Computer Science*, vol. 5753, pp. 637–654. Springer, Berlin-Heidelberg, Germany, 2009.

[DS99] Daniel De Schreye and Marc Denecker. Assessment of some issues in CL-theory and program development. In Krzysztof R. Apt, Victor Marek, Miroslaw Truszczynski, and David S. Warren (eds.), *The Logic Programming Paradigm: A 25 Years Perspective*, Artificial Intelligence Series, pp. 195–208. Springer, Berlin-Heidelberg, Germany, 1999.

[Eit99] Thomas Eiter, Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. The diagnosis frontend of the `dlv` system. *AI Communications*, 12(1–2):99–111, 1999.

[Eit05] Thomas Eiter, Hans Tompits, and Stefan Woltran. On solution correspondences in answer-set programming. In Leslie Pack Kaelbling and Alessandro Saffiotti (eds.), *Proceedings of the 19th International Joint Conference on Artificial Intelligence* (*IJCAI'05*), *Edinburgh, Scotland, UK, July 30-August 5, 2005*, pp. 97–102. Professional Book Center, Denver, CO, USA, 2005.

[Eit07] Thomas Eiter, Michael Fink, and Stefan Woltran. Semantical characterizations and complexity of equivalences in answer set programming. *ACM Transactions on Computational Logic*, 8(3):17, 2007.

[Erd09a] Esra Erdem. PHYLO-ASP: Phylogenetic systematics with answer set programming. In Esra Erdem, Fangzhen Lin, and Torsten Schaub (eds.), *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'09*), *Potsdam, Germany, September 14-18, 2009*, *Lecture Notes in Computer Science*, vol. 5753, pp. 567–572. Springer, Berlin-Heidelberg, Germany, 2009.

[Erd09b] Esra Erdem, Ozan Erdem, and Ferhan Türe. HAPLO-ASP: Haplotype inference using answer set programming. In Esra Erdem, Fangzhen Lin, and Torsten Schaub (eds.), *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'09*), *Potsdam, Germany, September 14-18, 2009*, *Lecture Notes in Computer Science*, vol. 5753, pp. 573–578. Springer, Berlin-Heidelberg, Germany, 2009.

[Fin08] Michael Fink. Equivalences in answer-set programming by countermodels in the logic of here-and-there. In Maria Garcia de la Banda and Enrico Pontelli (eds.), *Proceedings of the 24th International Conference on Logic Programming* (*ICLP'08*), *Udine, Italy, December 9-13, 2008*, *Lecture Notes in Computer Science*, vol. 5366, pp. 99–113. Springer, Berlin-Heidelberg, Germany, 2008.

[Geb08] Martin Gebser, Jörg Pührer, Torsten Schaub, and Hans Tompits. A meta-programming technique for debugging answer-set programs. In Dieter Fox and Carla P. Gomes (eds.), *Proceedings of the 23rd AAAI Conference on Artificial Intelligence* (*AAAI'08*), *Chicago, IL, USA, July 13-17, 2008*, pp. 448–453. AAAI Press, Menlo Park, CA, USA, 2008.

[Gel88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference on Logic Programming* (*ICLP'88*), *Seattle, WA, USA, August 15-19, 1988*, pp. 1070–1080. The MIT Press, 1988.

[Gel91a] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.

[Gel91b] Michael Gelfond and Halina Przymusińska. Definitions in epistemic specifications. In Wiktor Marek, Anil Nerode, and V. S. Subrahmanian (eds.), *Proceedings of the 1st International Workshop on Logic Programming and Non-monotonic Reasoning* (*LPNMR'91*), *Washington, D.C., USA, July 23, 1991*, pp. 245–259. MIT Press, Cambridge, MA, USA, 1991.

[Gel96] Michael Gelfond and Halina Przymusińska. Towards a theory of elaboration tolerance: Logic programming approach. *Journal on Software and Knowledge Engineering*, 6(1):89–112, 1996.

[Gel01] Michael Gelfond, Marcello Balduccini, and Joel Galloway. Diagnosing physical systems in A-prolog. In Thomas Eiter, Wolfgang Faber, and Mirosław Truszczyński (eds.), *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'01*), *Vienna, Austria, September 17-19, 2001*, *Lecture Notes in Computer Science*, vol. 2173, pp. 213–225. Springer, Berlin-Heidelberg, Germany, 2001.

[Gel02] Michael Gelfond. The USA-Advisor: A case study in answer set programming. In Sergio Flesca, Sergio Greco, Nicola Leone, and Giovambattista Ianni (eds.), *Proceedings of the 8th European Conference on Logics in Artificial Intelligence* (*JELIA'02*), *Cosenza, Italy, September, 23-26, 2002*, *Lecture Notes in Computer Science*, vol. 2424, pp. 566–568. Springer, Berlin-Heidelberg, Germany, 2002.

[Hel03] Keijo Heljanko and Ilkka Niemelä. Bounded LTL model checking with stable models. *Theory and Practice of Logic Programming*, 3(4-5):519–550, 2003.

[Het91] William C. Hetzel and Bill Hetzel. *The Complete Guide to Software Testing*. John Wiley & Sons, Inc., New York, NY, USA, 1991.

[Iel09] Salvatore Maria Ielpa, Salvatore Iiritano, Nicola Leone, and Francesco Ricca. An ASP-based system for e-tourism. In Esra Erdem, Fangzhen Lin, and Torsten Schaub (eds.), *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'09*), *Potsdam, Germany, September 14-18, 2009*, *Lecture Notes in Computer Science*, vol. 5753, pp. 368–381. Springer, Berlin-Heidelberg, Germany, 2009.

[Ino04] Katsumi Inoue and Chiaki Sakama. Equivalence of logic programs under updates. In José Júlio Alferes and João Alexandre Leite (eds.), *Proceedings of the 9th European Conference on Logics in Artificial Intelligence* (*JELIA'04*), *Lisbon, Portugal, September 27-30, 2004*, *Lecture Notes in Computer Science*, vol. 3229, pp. 174–186. Springer, Berlin-Heidelberg, Germany, 2004.

[Jan09] Tomi Janhunen, Emilia Oikarinen, Hans Tompits, and Stefan Woltran. Modularity aspects of disjunctive stable models. *Journal of Artificial Intelligence Research*, 35:813–857, 2009.

[Lif01] Vladimir Lifschitz, David Pearce, and Agustín Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.

[Mik07] Artur Mikitiuk, Eric Moseley, and Mirosław Truszczyński. Towards debugging of answer-set programs in the language PSpb. In *Proceedings of the 2007 International Conference on Artificial Intelligence* (*ICAI'07*), *Las Vegas, NV, USA, June 25-28, 2007*, vol. II, pp. 635–640. CSREA Press, 2007.

[Mil09] Alessandra Mileo, Davide Merico, and Roberto Bisiani. Non-monotonic reasoning supporting wireless sensor networks for intelligent monitoring: The SINDI system. In Esra Erdem, Fangzhen Lin, and Torsten Schaub (eds.), *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'09*), *Potsdam, Germany, September 14-18, 2009*, *Lecture Notes in Computer Science*, vol. 5753, pp. 585–590. Springer, Berlin-Heidelberg, Germany, 2009.

[Mye79] Glenford J. Myers. *The Art of Software Testing*. John Wiley & Sons, New York, NY, USA, 1979.

[Nie10] Ilkka Niemelä, Tomi Janhunen, Johannes Oetsch, Jörg Pührer, and Hans Tompits. On testing answer-set programs. In *Proceedings of the 19th European Conference on Artificial Intelligence* (*ECAI'10*), *Lisbon, Portugal, August 16-20*. 2010. To appear.

[Oet07] Johannes Oetsch, Hans Tompits, and Stefan Woltran. Facts do not cease to exist because they are ignored: Relativised uniform equivalence with answer-set projection. In *Proceedings of the 22nd Conference on Artificial Intelligence* (*AAAI'07*), *Vancouver, BC, Canada, July 22-26, 2007*, pp. 458–464. AAAI Press, Menlo Park, CA, USA, 2007.

[Oet08] Johannes Oetsch and Hans Tompits. Program correspondence under the answer-set semantics: The non-ground case. In Maria Garcia de la Banda and Enrico Pontelli (eds.), *Proceedings of the 24th International Conference on Logic Programming* (*ICLP'08*), *Udine, Italy, December 9-13, 2008*, *Lecture Notes in Computer Science*, vol. 5366, pp. 591–605. Springer, Berlin-Heidelberg, Germany, 2008.

[Oet10] Johannes Oetsch, Jörg Pührer, and Hans Tompits. Catching the Ouroboros: Towards debugging non-ground answer-set programs. *Theory and Practice of Logic Programming. Special Issue on the 2010 International Conference on Logic Programming*, 2010.

[Oik06] Emilia Oikarinen and Tomi Janhunen. Modular equivalence for normal logic programs. In Gerhard Brewka, Silvia Coradeschi, Anna Perini, and Paolo Traverso (eds.), *Proceedings of the 17th European Conference on Artificial Intelligence* (*ECAI'06*), *Riva del Garda, Italy, August 28-September 1, 2006*, pp. 412–416. IOS Press, Amsterdam, The Netherlands, 2006.

[Pal09] Alessandro Dal Palù, Agostino Dovier, and Enrico Pontelli. Logic programming techniques in protein structure determination: Methodologies and results. In Esra Erdem, Fangzhen Lin, and Torsten Schaub (eds.), *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'09*), *Potsdam, Germany, September 14-18, 2009*, *Lecture Notes in Computer Science*, vol. 5753, pp. 560–566. Springer, Berlin-Heidelberg, Germany, 2009.

[Per07] Simona Perri, Francesco Ricca, Giorgio Terracina, Daniela Cianni, and Pierfrancesco Veltri. An integrated graphic tool for developing and testing DLV programs. In Marina De Vos and Torsten Schaub (eds.), *Proceedings of the 1st International Workshop on Software Engineering for Answer Set Programming* (*SEA'07*), *Tempe, AZ, USA, May 14, 2007*, *CEUR Workshop Proceedings*, vol. 281, pp. 71–85. CEUR-WS.org, Aachen, Germany, 2007.

[Pon09] Enrico Pontelli, Tran Cao Son, and Omar El-Khatib. Justifications for logic programs under answer set semantics. *Theory and Practice of Logic Programming*, 9(1):1–56, 2009.

[Sur07] Adrian Sureshkumar, Marina De Vos, Martin Brain, and John Fitch. APE: An AnsProlog* environment. In Marina De Vos and Torsten Schaub (eds.), *Proceedings of the 1st International Workshop on Software Engineering for Answer Set Programming* (*SEA'07*), *Tempe, AZ, USA, May 14, 2007*, *CEUR Workshop Proceedings*, vol. 281, pp. 71–85. CEUR-WS.org, Aachen, Germany, 2007.

[Syr06] Tommi Syrjänen. Debugging inconsistent answer-set programs. In *Proceedings of the 11th International Workshop on Nonmonotonic Reasoning* (*NMR'06*), *Lake District, U.K., May 30-June 1, 2006*, *IfI Technical Report Series*, vol. IfI-06-04, pp. 77–83. Institut für Informatik, Technische Universität Clausthal, Clausthal-Zellerfeld, Germany, 2006.

[Tru09] Mirosław Truszczyński and Stefan Woltran. Relativized hyperequivalence of logic programs for modular programming. *Theory and Practice of Logic Programming*, 9(6):781–819, 2009.

[Wit09] Johan Wittocx, Hanne Vlaeminck, and Marc Denecker. Debugging for model expansion. In Patricia M. Hill and David Scott Warren (eds.), *Proceedings of the 25th International Conference on Logic Programming* (*ICLP'09*), *Pasadena, CA, USA, July 14-17, 2009*, *Lecture Notes in Computer Science*, vol. 5649, pp. 296–311. Springer, Berlin-Heidelberg, Germany, 2009.