Towards Quantified Answer Set Programming

Giovanni Amendola

Department of Mathematics and Computer Science, University of Calabria, Italy
amendola@mat.unical.it

Abstract. This paper proposes an extension of Answer Set Programming (ASP) with quantifiers called Quantified ASP (QASP). This proposal is somehow inspired to the extension of SAT formulas in Quantified Boolean Formulas (QBF). The resulting language is more expressive than plain ASP (which remains a subclass of QASP), and allows modeling problems belonging to any level of the polynomial hierarchy.

Keywords: Logic Programming · Answer Set Programming · Quantified Boolean Formulas.

1 Introduction

Quantified boolean formulas (QBF) are a powerful generalization of the satisfiability problem (SAT) in which the variables are also allowed to be universally as well as existentially quantified [34, 30]. The ability to nest universal and existential quantification makes QBF very expressive (any problem in the polynomial hierarchy can be encoded in QBF), and strictly more expressive than SAT which is bounded to NP complete problems. Solving QBF formulas has become an attractive and important research area over the last years. Indeed, many important artificial intelligence (AI) problems, such as planning, non monotonic reasoning, scheduling, model checking and verification formal verification, can be reduced to QBF [20, 41, 37]. So that, several solvers for QBF formulas have been proposed [31, 9, 42, 36].

Answer Set Programming (ASP) [18, 23, 16] is a well-established formalism for nonmonotonic reasoning, and combines a comparatively high knowledge-modeling power [18, 2] with a robust solving technology [19, 24–27, 35, 9, 13–15, 40, 39], that can also be responsive in case of incoherent logic programs [3, 11, 10, 6, 7]. For these reasons ASP has become an established logic-based programming paradigm with successful applications to complex problems in Artificial Intelligence [33, 32, 8, 5, 4], Databases [38], Game Theory [12], Information Extraction [1], and many other fields of knowledge.

ASP is very related to SAT [29]. Indeed, non-disjunctive ASP can model in a natural declarative way NP-hard combinatorial problems by encoding them as a logic program and computing its answer sets, which encode the problem solutions. Once disjunctive rules are admitted in an ASP programs, ASP can be used to model (often in a rather difficult and unnatural way) problems belonging to the second level of the polynomial hierarchy at most [21, 22].

Inspired by the way SAT is extended in QBF, in this paper we propose an extension of ASP with quantifiers that is called QASP (Quantified ASP). The ability to nest

universal and existential quantification makes QASP strictly more expressive than ASP. Indeed, we prove in the paper that QBF can be reduced naturally to QASP, thus any problem in the polynomial hierarchy can be also encoded in QASP. QASP combines the quantifiers of QBF with the non-monotonic reasoning features of ASP.

2 Preliminaries

We start with recalling syntax and semantics of answer set programming. We concentrate on logic programs over a propositional signature Σ. A *disjunctive rule* r is of the form

$$a_1 \vee \cdots \vee a_l \leftarrow b_1, ..., b_m, not\ c_1, ..., not\ c_n, \tag{1}$$

where all a_i, b_j, and c_k are atoms (from Σ); $l, m, n \geq 0$, and $l + m + n > 0$; *not* represents *negation-as-failure*. The set $H(r) = \{a_1, ..., a_l\}$ is the *head* of r, while $B^+(r) = \{b_1, ..., b_m\}$ and $B^-(r) = \{c_1, \ldots, c_n\}$ are the *positive body* and the *negative body* of r, respectively; the *body* of r is $B(r) = B^+(r) \cup B^-(r)$. We denote by $At(r) = H(r) \cup B(r)$ the set of all atoms occurring in r. A rule r is a *fact*, if $B(r) = \emptyset$ (we then omit $\leftarrow$); a *constraint*, if $H(r) = \emptyset$; *normal*, if $|H(r)| \leq 1$ and *positive*, if $B^-(r) = \emptyset$. A *(disjunctive logic) program* P is a finite set of disjunctive rules. P is called *normal* [resp. *positive*] if each $r \in P$ is normal [resp. positive]. We set $At(P) = \bigcup_{r \in P} At(r)$, that is the set of all atoms occurring in the program P.

Any subset I of Σ is an *interpretation*. An interpretation I is a *model* of a program P (denoted $I \models P$) if and only if for each rule $r \in P$, $I \cap H(r) \neq \emptyset$ if $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$ (denoted $I \models r$). A model M of P is *minimal*, if and only if there is no model M' of P such that $M' \subset M$. We denote by $MM(P)$ the set of all minimal models of P. Given an interpretation I, let P^I be the well-known *Gelfond-Lifschitz reduct* [28] of P with respect to I, i.e., the set of rules $a_1 \vee ... \vee a_l \leftarrow b_1, ..., b_m$, obtained from rules $r \in P$ of form (1), such that $B^-(r) \cap I = \emptyset$. An interpretation I is an *answer set* of P if $I \in MM(P^I)$. We denote by $AS(P)$ the set of all answer sets (called also *stable models*) of P.

Example 1. Consider the following logic program

$$P = \{a \leftarrow not\ b;\ b \leftarrow c,\ not\ d;\ c \leftarrow a;\ d \leftarrow not\, b\}.$$

For instance, a model of P is $\{b, d\}$. Moreover, the set of all minimal models of P is given by $MM(P) = \{\{b\}, \{a, c, d\}\}$. Now, let $I = \{a, c, d\}$. The Gelfond-Lifschitz reduct of P with respect to I is $P^I = \{a;\ c \leftarrow a;\ d\}$. Thus, $\{a, c, d\}$ is a minimal model of P^I. Hence, it is an answer set of P. On the other hand, $\{b\}$ is not an answer set of P. Indeed, $P^{\{b\}} = \{b \leftarrow c;\ c \leftarrow a\}$, but $\{b\}$ is not a minimal model of $P^{\{b\}}$ (as $MM(P^{\{b\}}) = \{\emptyset\}$).

3 Quantified Answer Set Programming

In this section, we introduce the syntax and the semantics of *Quantified Answer Set Programming* (QASP). Intuitively, we consider a partition of the atoms of a logic program, identifying different levels and quantifications (existential or universal) for each atom.

Then, we show that QASP semantics extends the standard ASP semantics. Finally, we note that modelling capabilities of QASP programs are beyond NP for normal QASP programs.

A QASP QP over a set of existential atoms $E = \{e_1, \ldots, e_n\}$ and a set of universal atoms $U = \{u_1, \ldots, u_m\}$, is a structure of the form:

$$Q_1 V_1 \ldots Q_k V_k P$$

where P is a logic program such that $At(P) = E \cup U$; $Q_i \in \{\exists, \forall\}$, for $i = 1, \ldots, k$; $V_i \subseteq E$ if $Q_i = \exists$, and $V_i \subseteq U$ if $Q_i = \forall$, for each $i = 1, \ldots, k$; and for each $i \neq j \in \{1, \ldots, k\}$, $V_i \cap V_j = \emptyset$, and $\bigcup_{i=1}^{k} V_i = E \cup U$. In the following, without loss of generality, we assume that $Q_i \neq Q_{i+1}$, for each $i = 1, \ldots, k-1$, i.e. we consider an alternation of existential and universal quantifiers, and moreover we assume that $V_1 = \forall$ and $V_k = \exists$. Note that to obtain a QASP program starting with an existential quantifier or ending with a universal quantifier, it is enough to set $V_1 = \emptyset$ or $V_k = \emptyset$, respectively. Therefore, we will consider QASP programs of the form $\forall U_1 \exists E_1 \ldots \forall U_h \exists E_h P$, where $U_1, \ldots, U_h \subseteq U$ and $E_1, \ldots, E_h \subseteq E$. We say that QP is a *normal* QASP program if P is normal.

Let $E' \subseteq E$, $U' \subseteq U$, and let I be a set of atoms. If $I \subseteq E'$, then I is called *partial existential interpretation* w.r.t. E', and if $I \subseteq U'$, then I is called *partial universal interpretation* w.r.t. U'. A *total interpretation* M of a QASP program QP is a forest (i.e., a disjoint union of trees), $M = \langle N, A \rangle$, where each node $n \in N$ is a pair $n = (I, J)$, where I is a partial universal interpretation w.r.t. $U_i \subseteq U$, and J is a partial existential interpretation w.r.t. $E_i \subseteq E$, for some $i = 1, \ldots, h$, that is called the *level* of the node n. At level 1 there exist exactly $2^{|U_1|}$ nodes $n = (I, J)$ such that I varies over all subsets of U_1. Moreover, for each node $n = (I, J)$ at level $i < h$, there exist exactly $2^{|U_{i+1}|}$ edges (n, n') such that $n' = (I', J')$ is a node at level $i+1$, where I' varies over all subsets of U_{i+1}.

A total interpretation M is a *quantified answer set* of a QASP program QP, if each path from a node at level 1 to a node at level h, that is a sequence of h nodes, $(I_1, J_1), \ldots, (I_h, J_h)$, is such that $I_1 \cup J_1 \cup \cdots \cup I_h \cup J_h$ is a minimal model of $P^{I_1 \cup J_1 \cup \cdots \cup I_h \cup J_h}$. We denote by $QAS(QP)$ the set of all quantified answer sets of QP.

Example 2. Consider the following QASP program:

$$QP = \forall U_1 \exists E_1 \forall U_2 \exists E_2 P,$$

over $E = \{e_1, e_2, e_3\}$ and $U = \{u_1, u_2\}$, where $U_1 = \emptyset$, $E_1 = \{e_1, e_2\}$, $U_2 = \{u_1, u_2\}$, $E_2 = \{e_3, e_4\}$, and

$$P = \left\{ \begin{array}{lll} e_3 \vee u_1 \vee u_2 \leftarrow e_1; & & \\ e_1 \leftarrow not\ u_1; & e_1 \leftarrow not\ e_3; & \\ e_4 \leftarrow u_1,\ not\ u_2; & e_4 \leftarrow u_2,\ not\ u_1; & \\ u_1 \leftarrow not\ e_3, not\ e_4; & u_2 \leftarrow not\ e_3, not\ e_4 & \end{array} \right\}.$$

Then, the graph reported in Figure 1 is a total interpretation of QP, where $(\emptyset, \{e_1\})$ is at level 1, and $(\emptyset, \{e_3\})$, $(\{u_1\}, \{e_4\})$, $(\{u_2\}, \{e_4\})$, and $(\{u_1, u_2\}, \emptyset)$ are at level 2. Moreover, it is also a quantified answer set. Indeed, $\{e_1, e_3\}$ is a minimal model of $P^{\{e_1, e_3\}} = \{e_3 \vee u_1 \vee u_2 \leftarrow e_1;\ e_1;\ e_4 \leftarrow u_1;\ e_4 \leftarrow u_2\}$; $\{e_1, u_1, e_4\}$ is a minimal model of $P^{\{e_1, u_1, e_4\}} = \{e_3 \vee u_1 \vee u_2 \leftarrow e_1;\ e_1;\ e_4 \leftarrow u_1\}$; $\{e_1, u_2, e_4\}$ is a minimal model of $P^{\{e_1, u_2, e_4\}} = \{e_3 \vee u_1 \vee u_2 \leftarrow e_1;\ e_1;\ e_4 \leftarrow u_2\}$; and $\{e_1, u_1, u_2\}$ is a minimal model of $P^{\{e_1, u_1, u_2\}} = \{e_3 \vee u_1 \vee u_2 \leftarrow e_1;\ e_1;\ u_1;\ u_2\}$.

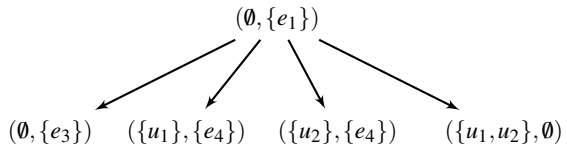

Fig. 1: Total interpretation of the program of the Example 2.

Intuitively, the notion of quantified answer set can be considered as an extension of the notion of answer set. Indeed, quantified answer sets coincides with answer sets whenever the QASP program contains existential atoms only, as stated in the following theorem.

Theorem 1. *Let P be a logic program, and let $QP = \forall U \exists E P$, where $U = \emptyset$ and $E = At(P)$. Then, $M \in AS(P)$ if, and only if, $(\emptyset, M) \in QAS(QP)$.*

Given the possibility of alternating universal and existential quantifiers, it is easy to see that the problem of deciding the existence of a quantified answer set such as its modelling capability are beyond NP problems.

Theorem 2. *To decide the existence of a quantified answer set for a QASP program is a PSPACE-complete problem, also for normal QASP program.*

4 Modeling Quantified Boolean Formulas in QASP

Let $\mathbf{X} = \{x_1, \ldots, x_k\}$ and $\mathbf{Y} = \{y_1, \ldots, y_n\}$ be two sets of variables. A Quantified Boolean Formula (QBF) is of the form $\Phi = \exists \mathbf{X} \forall \mathbf{Y} \phi(X,Y)$, where $\phi(X,Y)$ is a disjunction of m clauses over variables in $\mathbf{X}$ and $\mathbf{Y}$, i.e., $\phi(\mathbf{X},\mathbf{Y}) = C_1 \vee \cdots \vee C_m$, and each clause C_j is the conjunction of three literals, i.e. $C_j = l_{1j} \wedge l_{2j} \wedge l_{3j}$, for $j = 1, \ldots, m$, such that $l_{kj} \in \mathbf{X} \cup \{\neg x \mid x \in \mathbf{X}\} \cup \mathbf{Y} \cup \{\neg y \mid y \in \mathbf{Y}\}$.

We can model the QBF formula Φ by the QASP program $QP_\Phi = \forall U_1 \exists E_1 \forall U_2 \exists E_2 P$, where $U_1 = \emptyset$, $E_1 = \mathbf{X}$, $U_2 = \mathbf{Y}$, $E_2 = \{g, nx_1, \ldots, nx_k, ny_1, \ldots, ny_n\}$, and P is the set of the following rules

$$
\begin{array}{ll}
x \vee nx & \text{for each } x \in \mathbf{X}; \\
y \vee ny & \text{for each } y \in \mathbf{Y}; \\
\quad g \leftarrow \sigma(l_{1j}) \wedge \sigma(l_{2j}) \wedge \sigma(l_{3j}) & \text{for each } j = 1, \ldots, m; \\
\quad g \leftarrow \mathit{not}\ g &
\end{array}
$$

where $\sigma(l_{kj})$ is equal to l_{kj}, if $l_{kj} \in \mathbf{X} \cup \mathbf{Y}$, is equal to nx, if $l_k j = \neg x$ for some $x \in \mathbf{X}$, and is equal to ny, if $l_k j = \neg y$ for some $y \in \mathbf{Y}$.

Theorem 3. *Let Φ be a QBF formula. Then, Φ is valid if, and only if, $QAS(QP_\Phi) \neq \emptyset$.*

Proof. Let Φ be a valid QBF formula. Then, there is a truth assignment π of the variables in $\mathbf{X}$ such that for each truth assignment μ of the variables in $\mathbf{Y}$, at least

one clause, say C_j, is satisfied. Let $M = \pi(\mathbf{X}) \cup \mu(\mathbf{Y})$ be the model satisfying C_j. Hence, $\sigma(l_{1j}) \wedge \sigma(l_{2j}) \wedge \sigma(l_{3j})$ is satisfied by $\sigma(M) = \{\sigma(l) \mid l \in M\}$. Therefore, $M' = \sigma(M) \cup \{g\}$ is a minimal model of $P^{M'}$. Thus, there exists a subset J_1 of E_1, namely $J_1 = \pi(\mathbf{X}) \cap \mathbf{X}$, such that for each subset I_2 of U_2, namely $I_2 = \mu(\mathbf{Y}) \cap \mathbf{Y}$, there is a subset J_2 of E_2, namely $J_2 = \{g\} \cup \sigma(\pi(\mathbf{X}) \cap \{\neg x \mid x \in \mathbf{X}\}) \cup \sigma(\mu(\mathbf{Y}) \cap \{\neg y \mid y \in \mathbf{Y}\})$ such that $J_1 \cup I_2 \cup J_2$ is a minimal model of $P^{J_1 \cup I_2 \cup J_2}$. Therefore, QP_Φ has a quantified answer set. Now assume that $QAS(QP_\Phi) \neq \emptyset$. Hence, it can be easily checked that a quantified answer set of QP_Φ satisfies Φ. Indeed, each subset of U_2 corresponds to an assignment of the universal variables of Φ, and each subset of E_1 corresponds to an assignment of the existential variables of Φ.

5 Future work

Inspired by the way SAT is extended in QBF, in this paper we propose an extension of ASP with quantifiers that is called QASP (Quantified ASP). At the moment this is a proposal of extension that we would like to discuss at the workshop to get feedback from the community. Mostly it will be the basis for future work. Where we will investigate the possibility of extending QASP problems with variables, as it is done in ASP. Moreover we will study the expressivity of the language by applying it to a number of problems, and we will compare it with existing formalism in this respect. In particular, we will assess the practical modeling advantages of QASP with respect to QBF and QCSP. Finally, the implementation of the language in a solver is also subject of future work. In particular, we will investigate how reasoning techniques and the various structural properties useful for solving problems in QBF and QCSP (see, e.g. [17]) can be adapted to build efficient QASP solvers.

References

1. Adrian, W.T., Manna, M., Leone, N., Amendola, G., Adrian, M.: Entity set expansion from the web via ASP. In: ICLP (Technical Communications). OASICS, vol. 58, pp. 1:1–1:5. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
2. Alviano, M., Amendola, G., Peñaloza, R.: Minimal undefinedness for fuzzy answer sets. In: AAAI 2017. pp. 3694–3700 (2017)
3. Amendola, G.: Dealing with incoherence in ASP: split semi-equilibrium semantics. In: DWAI@AI*IA. CEUR Workshop Proceedings, vol. 1334, pp. 23–32. CEUR-WS.org (2014)
4. Amendola, G.: Preliminary results on modeling interdependent scheduling games via answer set programming. In: RCRA@AI*IA. p. to appear. CEUR Workshop Proceedings, CEUR-WS.org (2018)
5. Amendola, G.: Solving the stable roommates problem using incoherent answer set programs. In: RCRA@AI*IA. p. to appear. CEUR Workshop Proceedings, CEUR-WS.org (2018)
6. Amendola, G., Dodaro, C., Faber, W., Leone, N., Ricca, F.: On the computation of paracoherent answer sets. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA. pp. 1034–1040 (2017)
7. Amendola, G., Dodaro, C., Faber, W., Ricca, F.: Externally supported models for efficient computation of paracoherent answer sets. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, February 2-7, 2018, New Orleans, Louisiana, USA. pp. 1034–1040 (2018)

8. Amendola, G., Dodaro, C., Leone, N., Ricca, F.: On the application of answer set programming to the conference paper assignment problem. In: AI*IA. Lecture Notes in Computer Science, vol. 10037, pp. 164–178. Springer (2016)
9. Amendola, G., Dodaro, C., Ricca, F.: ASPQ: an asp-based 2qbf solver. In: QBF@SAT. CEUR Workshop Proceedings, vol. 1719, pp. 49–54. CEUR-WS.org (2016)
10. Amendola, G., Eiter, T., Fink, M., Leone, N., Moura, J.: Semi-equilibrium models for paracoherent answer set programs. Artif. Intell. **234**, 219–271 (2016)
11. Amendola, G., Eiter, T., Leone, N.: Modular paracoherent answer sets. In: Logics in Artificial Intelligence - 14th European Conference, JELIA2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings. pp. 457–471 (2014)
12. Amendola, G., Greco, G., Leone, N., Veltri, P.: Modeling and reasoning about NTU games via answer set programming. In: IJCAI 2016. pp. 38–45 (2016)
13. Amendola, G., Ricca, F., Truszczynski, M.: Generating hard random boolean formulas and disjunctive logic programs. In: IJCAI. pp. 532–538. ijcai.org (2017)
14. Amendola, G., Ricca, F., Truszczynski, M.: A generator of hard 2qbf formulas and asp programs. In: KR. AAAI Press (2018)
15. Amendola, G., Ricca, F., Truszczynski, M.: Random models of very hard 2qbf and disjunctive programs: An overview. In: ICTCS. CEUR Workshop Proceedings, CEUR-WS.org (2018)
16. Bonatti, P.A., Calimeri, F., Leone, N., Ricca, F.: Answer set programming. In: 25 Years GULP. Lecture Notes in Computer Science, vol. 6125, pp. 159–182. Springer (2010)
17. Bordeaux, L., Cadoli, M., Mancini, T.: Generalizing consistency and other constraint properties to quantified constraints. ACM Trans. Comput. Log. **10**(3), 17:1–17:25 (2009)
18. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Com. ACM **54**(12), 92–103 (2011)
19. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: Design and results of the Fifth Answer Set Programming Competition. Artif. Intell. **231**, 151–181 (2016)
20. Cashmore, M., Fox, M., Giunchiglia, E.: Planning as quantified boolean formula. In: ECAI. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 217–222. IOS Press (2012)
21. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: Propositional case. Ann. Math. Artif. Intell. **15**(3-4), 289–323 (1995)
22. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive datalog. ACM Trans. Database Syst. **22**(3), 364–418 (1997)
23. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Morgan & Claypool Publishers (2012)
24. Gebser, M., Leone, N., Maratea, M., Perri, S., Ricca, F., Schaub, T.: Evaluation techniques and systems for answer set programming: a survey. In: IJCAI 2018. pp. 5450–5456 (2018)
25. Gebser, M., Maratea, M., Ricca, F.: The Design of the Sixth Answer Set Programming Competition. In: LPNMR. LNCS, vol. 9345, pp. 531–544. Springer (2015)
26. Gebser, M., Maratea, M., Ricca, F.: What's hot in the answer set programming competition. In: AAAI. pp. 4327–4329. AAAI Press (2016)
27. Gebser, M., Maratea, M., Ricca, F.: The sixth answer set programming competition. Journal of Artificial Intelligence Research **60**, 41–95 (2017)
28. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. **9**(3/4), 365–386 (1991). https://doi.org/10.1007/BF03037169, `http://dx.doi.org/10.1007/BF03037169`
29. Giunchiglia, E., Lierler, Y., Maratea, M.: Sat-based answer set programming. In: AAAI. pp. 61–66. AAAI Press / The MIT Press (2004)
30. Giunchiglia, E., Marin, P., Narizzano, M.: Reasoning with quantified boolean formulas. In: Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 761–780. IOS Press (2009)

31. Giunchiglia, E., Narizzano, M., Tacchella, A.: QUBE: A system for deciding quantified boolean formulas satisfiability. In: IJCAR. Lecture Notes in Computer Science, vol. 2083, pp. 364–369. Springer (2001)
32. Grasso, G., Iiritano, S., Leone, N., Lio, V., Ricca, F., Scalise, F.: An asp-based system for team-building in the gioia-tauro seaport. In: PADL. Lecture Notes in Computer Science, vol. 5937, pp. 40–42. Springer (2010)
33. Grasso, G., Iiritano, S., Leone, N., Ricca, F.: Some DLV applications for knowledge management. In: LPNMR. Lecture Notes in Computer Science, vol. 5753, pp. 591–597. Springer (2009)
34. Kleine Büning, H., Bubeck, U.: Theory of quantified boolean formulas. In: Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 735–760. IOS Press (2009)
35. Lierler, Y., Maratea, M., Ricca, F.: Systems, engineering environments, and competitions. AI Magazine **37**(3), 45–52 (2016)
36. Lonsing, F., Egly, U.: Depqbf 6.0: A search-based QBF solver beyond traditional QCDCL. In: CADE. Lecture Notes in Computer Science, vol. 10395, pp. 371–384. Springer (2017)
37. Mangassarian, H., Veneris, A.G., Benedetti, M.: Robust QBF encodings for sequential circuits with applications to verification, debug, and test. IEEE Trans. Computers **59**(7), 981–994 (2010)
38. Manna, M., Ricca, F., Terracina, G.: Taming primary key violations to query large inconsistent data via ASP. TPLP **15**(4-5), 696–710 (2015)
39. Maratea, M., Pulina, L., Ricca, F.: A multi-engine approach to answer-set programming. TPLP **14**(6), 841–868 (2014)
40. Maratea, M., Ricca, F., Faber, W., Leone, N.: Look-back techniques and heuristics in DLV: implementation, evaluation, and comparison to QBF solvers. J. Algorithms **63**(1-3), 70–89 (2008)
41. Marin, P., Narizzano, M., Pulina, L., Tacchella, A., Giunchiglia, E.: Twelve years of QBF evaluations: QSAT is pspace-hard and it shows. Fundam. Inform. **149**(1-2), 133–158 (2016)
42. Rabe, M.N., Tentrup, L.: CAQE: A certifying QBF solver. In: FMCAD. pp. 136–143. IEEE (2015)