Computing Modular Paracoherent Answer Sets: Preliminary Result

Bernardo Cuteri[1], Carmine Dodaro[2], and Francesco Ricca[1]

[1] University of Calabria, Italy - lastname@mat.unical.it
[2] University of Genoa, Italy - dodaro@dibris.unige.it

Abstract. Answer Set Programming (ASP) is a well-established logic programming formalism. Problem solving in ASP requires to write an ASP program whose answers sets correspond to solutions. Despite the non-existence of answer sets for some ASP programs can be a modeling feature, it turns out to be a weakness in many other cases, and especially for query answering. Paracoherent answer set semantics extend the classical semantics of ASP to draw meaningful conclusions also from incoherent logic programs, with the result of increasing the range of applications of ASP. State of the art implementations of paracoherent ASP adopt the semi-stable and the semi-equilibrium models semantics, but cannot compute split semi-equilibrium semantics, also known as modular paracoherent semantics, that discards undesirable semi-equilibrium models. In this paper, we introduce and discuss a first approach for the computation of modular paracoherent answer sets.

Keywords: Answer Set Programming · Paracoherent Reasoning · Semi-equilibrium models

1 Introduction

Answer Set Programming (ASP) [22,23,34] is a well-established logic programming language, based on the stable model (or answer set) semantics [40], with a robust solving technology [43,3,24,35,36,37,39,12,16,17,18,47,46,38,42,41]. As a matter of fact, ASP has been applied to solve complex problems in several areas of knowledge such as AI [21,31,32,11,1,7]; Bioinformatics [25]; Databases [19,45]; Game Theory [15,6]; and industrial applications [27,28].

The non-existence of answer sets for some ASP programs can be a modeling feature, but, as argued in [13], it turns out to be a weakness in many other applications, such as: debugging, model building, inconsistency-tolerant query answering, diagnosis, planning and reasoning about actions. To remedy to the non-existence of answer sets, paracoherent semantics extend the classical answer set semantics to draw meaningful conclusions also from incoherent programs. This ASP variant has been termed *paracoherent reasoning* [13]. In particular, [13] improved the paracoherent semantics of *semi-stable models* [51] avoiding some anomalies with respect to basic modal logic properties by resorting to equilibrium

logic [48]. Thus, this paracoherent semantics is called *semi-equilibrium model (SEQ) semantics* [13].

More recently, [13] noticed that, although the SEQ semantics has nice properties, it may select models that do not respect modular structure of the program. SEQ semantics use 3-valued interpretations where a third truth value besides *true* and *false* expresses that an atom is *believed true*. For instance, the incoherent logic program $P = \{b \leftarrow not\ a;\ c \leftarrow not\ a, not\ c\}$ admits two SEQ models, say M_1 and M_2. In M_1, b is true, c is believed true, and a is false; whereas in M_2 a is believed true and both b and c are false. Now, M_1 appears preferable to M_2, as, according with a layering (stratification) principle, which is widely agreed in logic programming, one should prefer b rather than a, as there is no way to derive a (note that a does not appear in the head of any rule of the program). Therefore, [14] refine SEQ-models using splitting sequences [44], the major tool for modularity in modeling and evaluating answer set programs. In particular, the refined semantics, called *Split SEQ model semantics*, also known as *modular paracoherent semantics*, is able to discard model M_2.

The first efficient implementations of paracoherent semantics were proposed recently [8,10], but they only support semi-stable and semi-equilibrium semantics. Although the Split SEQ semantics discards some undesirable SEQ models, the existing methods for computing SEQ models are not able to compute the refined semantics. In this paper, we fill this lack presenting and discussing a first strategy for computing a modular paracoherent answer set.

2 Preliminaries

We start with recalling answer set semantics, and then present the paracoherent semantics of semi-equilibrium models, and its refined version based on splitting sequences.

2.1 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); $l > 0$, $m, n \geq 0$; *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 $\leftarrow$); *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 let $At(P) = \bigcup_{r \in P} At(r)$, that is the set of all atoms occurring in the program P.

The *dependency graph* of a program P is the directed graph $DG(P) = \langle V_P, E_P \rangle$ whose nodes V_P are the atoms in P and E_P contains an edge (a, b) if a occurs in $H(r)$ and either b occurs in $B(r)$ or in $H(r) \setminus \{a\}$. The *strongly connected components* (SCCs) of P, denoted $SCC(P)$, are the SCCs of $DG(P)$, which are the maximal sets of nodes C such that every pair of nodes is connected by some path in $DG(P)$ with nodes only from C.

Any set $I \subseteq \Sigma$ is an *interpretation*; it 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 no model $M' \subset M$ of P exists. We denote by $MM(P)$ the set of all minimal models of P and by $AS(P)$ the set of all *answer sets (or stable models)* of P, i.e., the set of all interpretations I such that $I \in MM(P^I)$, where P^I is the well-known *Gelfond-Lifschitz reduct* [40] of P w.r.t. 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 $B^-(r) \cap I = \emptyset$. We say that a program P is *coherent*, if it admits some answer set (i.e., $AS(P) \neq \emptyset$), otherwise, it is *incoherent*.

2.2 Semi-Equilibrium Models

Here, we introduce the paracoherent semantics of the *semi-equilibrium (SEQ) models* introduced in [13]. Consider an extended signature $\Sigma^\kappa = \Sigma \cup \{Ka \mid a \in \Sigma\}$. Intuitively, Ka can be read as a is believed to hold. The SEQ models of a program P are obtained from its *epistemic HT-transformation* P^{HT}, defined as follows.

Definition 1. *Let P be a program over Σ. Then its epistemic HT-transformation P^{HT} is obtained from P by replacing each rule r of the form (1) in P, such that $B^-(r) \neq \emptyset$, with:*

$$\lambda_{r,1} \vee \ldots \vee \lambda_{r,l} \vee Kc_1 \vee \ldots \vee Kc_n \leftarrow b_1, \ldots, b_m, \tag{2}$$

$$a_i \leftarrow \lambda_{r,i}, \tag{3}$$

$$\leftarrow \lambda_{r,i}, c_j, \tag{4}$$

$$\lambda_{r,i} \leftarrow a_i, \lambda_{r,k}, \tag{5}$$

for $1 \leq i, k \leq l$ and $1 \leq j \leq n$, where the $\lambda_{r,i}$, $\lambda_{r,k}$ are fresh atoms; and by adding the following set of rules:

$$Ka \leftarrow a, \tag{6}$$

$$Ka_1 \vee ... \vee Ka_l \vee Kc_1 \vee ... \vee Kc_n \leftarrow Kb_1, ..., Kb_m, \tag{7}$$

for $a \in \Sigma$, respectively for every rule $r \in P$ of the form (1).

Note that for any program P, its epistemic HT-transformation P^{HT} is positive. For every interpretation I^κ over $\Sigma' \supseteq \Sigma^\kappa$, let $\mathcal{G}(I^\kappa) = \{Ka \in I^\kappa \mid a \notin I^\kappa\}$ denote the atoms believed true but not assigned true, also referred to as the gap of I^κ. Given a set $\mathcal{F}$ of interpretations over Σ', an interpretation $I^\kappa \in \mathcal{F}$

is *maximal canonical in* $\mathcal{F}$, if no $J^\kappa \in \mathcal{F}$ exists such that $\mathcal{G}(I^\kappa) \supset \mathcal{G}(J^\kappa)$. By $mc(\mathcal{F})$ we denote the set of maximal canonical interpretations in $\mathcal{F}$. SEQ models are then defined as *maximal canonical* interpretations among the answer sets of P^{HT}.

Definition 2. *Let P be a program over Σ, and let I^κ be an interpretation over Σ^κ. Then, $I^\kappa \in SEQ(P)$ if, and only if, $I^\kappa \in \{M \cap \Sigma^\kappa \mid M \in mc(AS(P^{HT}))\}$, where $SEQ(P)$ is the set of semi-equilibrium models of P.*

2.3 Split Semi-equilibrium Models

A set $S \subseteq At(P)$ is a *splitting set* of P, if for every rule r in P such that $H(r) \cap S \neq \emptyset$ we have that $At(r) \subseteq S$. We denote by $b_S(P) = \{r \in P \mid At(r) \subseteq S\}$ the *bottom* part of P, and by $t_S(P) = P \setminus b_S(P)$ the *top* part of P relative to S. A *splitting sequence* $S = (S_1, \ldots, S_n)$ of P is a sequence of splitting sets S_i of P such that $S_i \subseteq S_j$ for each $i < j$. Let $SCC(P)$ be the set of all strongly connected components of P, and let $(C_1, \ldots, C_n)$ be a topological ordering of $SCC(P)$. It is known that $\Gamma = (\Gamma_1, \ldots, \Gamma_n)$, where $\Gamma_j = C_1 \cup \ldots \cup C_j$ for $j = 1, \ldots, n$, is a splitting sequence of P. So that, we obtain a *stratification* for P in subprograms $(P_1, \ldots, P_n)$ such that $P_1 = b_{\Gamma_1}(P)$, and $P_j = b_{\Gamma_j}(P) \setminus P_{j-1}$, for $j = 2, \ldots, n$. Given an interpretation M_i over C_i, we denote by $info(M_i)$ the set of rules $\{a \mid a \in M_i\} \cup \{\leftarrow not\ a \mid Ka \in M_i\} \cup \{\leftarrow a \mid a \in C_i \setminus M_i\}$.

Definition 3. *Given a topological ordering $(C_1, \ldots, C_n)$ of $SCC(P)$, an interpretation M over $At(P)$ is a* semi-equilibrium model of P relative to Γ *if there is a sequence of interpretations $M_1, \ldots, M_n$ over $\Gamma_1, \ldots \Gamma_n$, respectively, such that* (1) *$M = M_n$;* (2) *$M_1 \in SEQ(P_1)$;* (3) *$M_j \in SEQ(P_j \cup info(M_{j-1}))$, for $j = 2, \ldots, n$; and* (4) *M is maximal canonical among the interpretations over $At(P)$ satisfying conditions* (1), (2) *and* (3)*. The set of all semi-equilibrium model of P relative to Γ is denoted by $SEQ^\Gamma(P)$.*

Since $SEQ^\Gamma(P)$ is independent by the given topological ordering of $SCC(P)$ (see, Theorem 5 in [13]), the SCC-models of P have been defined as the set $M^{SCC}(P) = SEQ^\Gamma(P)$ for an arbitrary topological ordering of $SCC(P)$. We will refer to them as split semi-equlibrium models. Finally, note that $M^{SCC}(P) \subseteq SEQ(P)$.

Example 1. Consider the program

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

Then, $(\{a\}, \{b\}, \{c, d\})$ is a topological ordering of $SCC(P)$, so that $\Gamma = (\{a\}, \{a, b\}, \{a, b, c, d\})$ is a splitting sequence for P. Hence, $SEQ^\Gamma(P) = \{\{b, Kb, Kc\}\}$. Indeed $P_1 = b_{\Gamma_1}(P) = \emptyset$ and thus $SEQ(P_1) = \{\emptyset\}$. Then, $P_2 \cup info(\emptyset) = \{b \leftarrow not\ a,\ \leftarrow a\}$ and thus $SEQ(P_2 \cup info(\emptyset)) = \{\{b\}\}$. Finally, $P_3 \cup info(\{b\}) = \{d \leftarrow b, not\ c;\ c \leftarrow d;\ b;\ \leftarrow a\}$ and thus $SEQ(P_3 \cup info(\{b\})) = \{\{b, Kb, Kc\}\}$.

In the following, we will refer to semi-equilibrium models as *paracoherent answer sets*, and to split semi-equilibrium models as *modular* paracoherent answer sets.

3 On the Computation of Split Semi-Equilibrium Models

In this section, we describe a strategy to compute a split semi-equilibrium model by exploiting the computation of a semi-equilibrium model.

First, we consider a possible path that can be generated through the splitting sequence. Note that, each path leads to obtain a paracoherent answer set of the last program (i.e., $P_n \cup \mathit{info}(M_{n-1})$), as stated in the following theorem.

Theorem 1. *Let P be a program and let $(P_1, ..., P_n)$ be a stratification for P. Then, for each $i = 1, \ldots, n-1$, $M \in SEQ(P_i)$ implies $SEQ(P_{i+1} \cup \mathit{info}(M)) \neq \emptyset$.*

Proof. It is known that whenever P has a classical model, then $SEQ(P) \neq \emptyset$. Hence, it is enough to show that $P_{i+1} \cup \mathit{info}(M)$ admits a classical model. Recall that $\mathit{info}(M) = \{a. \mid a \in M\} \cup \{\leftarrow \mathit{not}\ a \mid Ka \in M\} \cup \{\leftarrow a \mid a \in C_i \setminus M\}$. Moreover, by construction of P_{i+1}, no atom appearing in a constraint of $\mathit{info}(M)$, appears in the head of a rule in P_{i+1}. Then, by setting as true each atom in the head of the rule in P_{i+1}, we obtain a classical model of $P_{i+1} \cup \mathit{info}(M)$.

Therefore, we could choose a paracoherent answer set M_1 of the first subprogram P_1 (that always exists as P_1 is constraint-free), then we move to find a model M_2 of the subprogram $P_2 \cup \mathit{info}(M_1)$ (that always exists by Theorem 1), and so on. At the end of the procedure, we obtain a paracoherent answer set M_n of the program $P_n \cup \mathit{info}(M_{n-1})$. However, it could not be a paracoherent answer set of the original program.

Example 2. Consider the program $P = \{a \leftarrow \mathit{not}\ b;\ b \leftarrow \mathit{not}\ a;\ c \leftarrow a,\ \mathit{not}\ c\}$. In the first layer of P, we have the subprogram $P_1 = \{a \leftarrow \mathit{not}\ b;\ b \leftarrow \mathit{not}\ a\}$ whose (paracoherent) answer sets are $\{a, Ka\}$ and $\{b, Kb\}$. So that, considering $\mathit{info}(\{a, Ka\}) \cup \{c \leftarrow a,\ \mathit{not}\ c\}$, we obtain the paracoherent answer set $\{a, Ka, Kc\}$, while considering $\mathit{info}(\{b, Kb\}) \cup \{c \leftarrow a,\ \mathit{not}\ c\}$, we obtain the (paracoherent) answer set $\{b, Kb\}$. Since $\mathcal{G}(\{a, Ka, Kc\}) \supset \mathcal{G}(\{b, Kb\})$, then $\{a, Ka, Kc\}$ cannot be a paracoherent answer set of P.

Intuitively, since each subprogram could have more than one answer set, we need to explore all possible paths by enumerating all possible paracoherent models obtainable from each path to make feasible a final phase of gap minimization.

More formally, let $(P_1, \ldots, P_n)$ be a stratification for a program P. We denote by PAS_1 the set of all semi-equilibrium models of P_1, i.e., $PAS_1 = SEQ(P_1)$; and, for each $i = 2, \ldots, n$, we denote by PAS_i the set of all semi-equilibrium models of $P_i \cup \mathit{info}(M_{i-1})$, where M_{i-1} varies among the semi-equilibrium models of PAS_{i-1}, i.e., $PAS_i = \{M \in SEQ(P_i \cup \mathit{info}(M_{i-1})) \mid M_{i-1} \in PAS_{i-1}\}$. Hence, the computation of a split semi-equilibrium model is given as follows.

(1) For each $i = 1, \ldots, n$, we compute PAS_i.
(2) Then, we look for a model M in PAS_n that is gap-minimal, with respect to subset inclusion, among all models in PAS_n.

Theorem 2. *Let P be a logic program. If $M \in PAS_n$ is gap-minimal with respect to subset inclusion, among all models in PAS_n, then M is a split semi-equilibrium model of P.*

Proof. Let $M \in PAS_n$. Hence, by definition of PAS_n, $M \in SEQ(P_n \cup info(M_{n-1}))$, for some model $M_{n-1} \in PAS_{n-1}$. Moreover, by definition of PAS_{n-1}, $M_{n-1} \in SEQ(P_{n-1} \cup info(M_{n-2}))$, for some model $M_{n-2} \in PAS_{n-2}$. By repeatedly applying the definition of PAS_i, for $i = 1, \ldots, n-1$, and considering models named M_i, at the end, we obtain that, by definition of PAS_2, $M_2 \in SEQ(P_2 \cup info(M_1))$, for some model $M_1 \in PAS_1 = SEQ(P_1)$. Hence, condition (2) and condition (3) in Definition 3 are satisfied. Finally, as M is gap-minimal with respect to subset inclusion, among all models in PAS_n, then M is maximal canonical among the interpretations over $At(P)$ satisfying the three conditions in Definition 3. Therefore, M is a split semi-equilibrium model of P.

Intuitively, this approach could provide an improvement in the practical computation of a (modular) paracoherent aswer set, as the search of a semi-equilibrium model is relative to small parts of the whole program. If n is sufficiently large, the size of each subprogram P_i, in the stratification of P, decreases enormously. In particular, the average of the size will be $\|P_i\| = \|P\|/n$, where $\|\cdot\|$ can be the number of atoms or the number of rules of a logic program. However, this observation must be counterbalanced by the fact, repeatedly observed, that it is now necessary to calculate all possible paths leading to a split semi-equilibrium model candidate, and these paths are in an exponential number. In fact, if for example we assume that the original program is stratified into three subprograms, say P_1, P_2 and P_3, such that each computation produces two semi-equilibrium models, we will have that: at step (1), $SEQ(P_1) = \{M_1^1, M_2^1\}$; at step (2), $SEQ(P_2 \cup info(M_1^1)) = \{M_1^2, M_2^2\}$ and $SEQ(P_2 \cup info(M_2^1)) = \{M_3^2, M_4^2\}$; and at step (3), $SEQ(P_3 \cup info(M_1^2)) = \{M_1^3, M_2^3\}$, $SEQ(P_3 \cup info(M_2^2)) = \{M_3^3, M_4^3\}$, $SEQ(P_3 \cup info(M_3^2)) = \{M_5^3, M_6^3\}$, $SEQ(P_3 \cup info(M_4^2)) = \{M_7^3, M_8^3\}$. Now, assume for instance that the computation time of a semi-equilibrium model of a program P is directly proportional to the size of P. We denote by $time(P)$ the time to compute a semi-equilibrium model of P. Hence, since we could compute an exponential number of semi-equilibrium models, say 2^n (like in the example), intuitively, the computation time will be about $time(P_1) \times 2^1 + \ldots + time(P_n) \times 2^n$ that will be proportional to $time(P) \times 2^1/n + \ldots + time(P) \times 2^n/n$, that is $time(P) \times \frac{2^{n+1}-1}{n}$. Therefore, it is desirable for the future to identify more sophisticated techniques, to try to overcome such a computational explosion.

4 Related Work

Semantics for non-monotonic logic programs [50,53,54,51,30,52,20,49,2,33,13,26] that relax the definition of answer set to overcome the absence of answer sets can be considered in broader terms paracoherent semantics. Nonetheless, the term paracoherent answer set was used for the first time by Inoue and Sakama in [51], where they introduced the semi-stable semantics as a remedy to the absence of answer sets due to cyclic negation. Later, in [14,5] some anomalies of semi-stable semantics with respect to some epistemic properties were evidenced, and the semi-equilibrium semantics was proposed as a remedy. In [13] it

was demonstrated that semi-equilibrium semantics features a number of highly desirable theoretical properties for a knowledge representation language (for instance, minimal undefinedness [4]), and at the same time, it was observed that semi-equilibrium models do not enjoy the same nice modular composition properties of stable models (e.g., the *splitting set* [44] modularity tool cannot be used straightforwardly). Notably, modular composition is used in ASP for simplifying the modeling of problems (actually, the guess and check programming methodology [29] is based on this property) and is a principle underlying the architectures of ASP systems [43]. The split semi-equilibrium semantics [13] solves this problem by using splitting sequences to decompose the program into hierarchically organized subprograms. Split semi-equilibrium models are semi equilibrium models that enjoy a modularity property.

Concerning the implementation of semi-stable and semi-equilibrium semantics, we observe that they have been implemented efficiently only recently. In particular, in [8] a number of algorithms has been proposed, that compute paracoherent answer sets in two steps: (*i*) an epistemic transformation of programs is applied, and (*ii*) a strategy for computing answer sets of minimum gap is implemented by calling (possibly multiple times) an ASP solver. The same strategy has been improved in [10] by replacing the classic epistemic transformations by more parsimonious ones (that we also adopt). The new transformations are based on the characterization of paracoherent answer sets in terms of *externally supported models*. Neither [8] nor [10] support split semi-equilibrium semantics that is the focus of this paper.

5 Conclusion

Paracoherent answer set semantics can draw meaningful conclusions also from incoherent programs, and in this way increase the applicability of ASP for solving AI problems [13]. Practical applications are possible once efficient implementations are available, and the complex task of computing efficiently a paracoherent answer set has been approached only recently [8,10,9]. State of the art solutions supported the semi-equilibrium semantics but cannot compute the split semi-equilibrium semantics. In this paper we presented a first approach to compute a split semi-equilibrium model.

As future work, we plan to identify more efficient evaluation strategies by exploiting the computational complexity properties of the modular paracoherent semantics, that could allow for computing a split semi-equilibrium model using a plain ASP solver. Finally, we will implement our approaches and compare them against existing implementations for semi-equilibrium models.

References

1. Adrian, W.T., Manna, M., Leone, N., Amendola, G., Adrian, M.: Entity set expansion from the web via ASP. In: ICLP-TC. OASICS, vol. 58, pp. 1:1–1:5 (2017)

2. Alcântara, J., Damásio, C.V., Pereira, L.M.: An encompassing framework for paraconsistent logic programs. J. Applied Logic **3**(1), 67–95 (2005)
3. Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., Ricca, F.: Evaluation of disjunctive programs in WASP. In: LPNMR. LNCS, vol. 11481, pp. 241–255. Springer (2019)
4. Alviano, M., Amendola, G., Peñaloza, R.: Minimal undefinedness for fuzzy answer sets. In: AAAI 2017. pp. 3694–3700 (2017)
5. 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)
6. Amendola, G.: Preliminary results on modeling interdependent scheduling games via answer set programming. In: RiCeRcA@AI*IA. CEUR Workshop Proceedings, vol. 2272 (2018)
7. Amendola, G.: Solving the stable roommates problem using incoherent answer set programs. In: RiCeRcA@AI*IA. CEUR Workshop Proceedings, vol. 2272 (2018)
8. Amendola, G., Dodaro, C., Faber, W., Leone, N., Ricca, F.: On the computation of paracoherent answer sets. In: AAAI'17. pp. 1034–1040 (2017)
9. Amendola, G., Dodaro, C., Faber, W., Pulina, L., Ricca, F.: Algorithm selection for paracoherent answer set computation. In: JELIA. LNCS, vol. 11468, pp. 479–489. Springer (2019)
10. Amendola, G., Dodaro, C., Faber, W., Ricca, F.: Externally supported models for efficient computation of paracoherent answer sets. In: AAAI. pp. 1720–1727. AAAI Press (2018)
11. Amendola, G., Dodaro, C., Leone, N., Ricca, F.: On the application of answer set programming to the conference paper assignment problem. In: AI*IA. LNCS, vol. 10037, pp. 164–178. Springer (2016)
12. 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)
13. Amendola, G., Eiter, T., Fink, M., Leone, N., Moura, J.: Semi-equilibrium models for paracoherent answer set programs. Artif. Intell. **234**, 219–271 (2016)
14. Amendola, G., Eiter, T., Leone, N.: Modular paracoherent answer sets. In: JELIA'14. pp. 457–471 (2014)
15. Amendola, G., Greco, G., Leone, N., Veltri, P.: Modeling and reasoning about NTU games via answer set programming. In: IJCAI'16. pp. 38–45 (2016)
16. Amendola, G., Ricca, F., Truszczynski, M.: Generating hard random boolean formulas and disjunctive logic programs. In: IJCAI. pp. 532–538. ijcai.org (2017)
17. Amendola, G., Ricca, F., Truszczynski, M.: A generator of hard 2qbf formulas and asp programs. In: KR. AAAI Press (2018)
18. Amendola, G., Ricca, F., Truszczynski, M.: Random models of very hard 2qbf and disjunctive programs: An overview. In: ICTCS. CEUR Workshop Proceedings (2018)
19. Arenas, M., Bertossi, L.E., Chomicki, J.: Answer sets for consistent query answering in inconsistent databases. TPLP **3**(4-5), 393–424 (2003)
20. Balduccini, M., Gelfond, M.: Logic programs with consistency-restoring rules. In: ISLFCR, AAAI'03. pp. 9–18 (2003)
21. Balduccini, M., Gelfond, M., Watson, R., Nogueira, M.: The usa-advisor: A case study in answer set planning. In: LPNMR. LNCS, vol. 2173, pp. 439–442. Springer (2001)
22. Baral, C.: Knowledge Representation, Reasoning, and Declarative Problem Solving. Cambridge University Press, New York, NY, USA (2003)

23. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Com. ACM **54**(12), 92–103 (2011)
24. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: Design and results of the fifth answer set programming competition. Artif. Intell. **231**, 151–181 (2016)
25. Campeotto, F., Dovier, A., Pontelli, E.: A declarative concurrent system for protein structure prediction on GPU. J. Exp. Theor. Artif. Intell. **27**(5), 503–541 (2015)
26. Costantini, S., Formisano, A.: Query answering in resource-based answer set semantics. TPLP **16**(5-6), 619–635 (2016)
27. Dodaro, C., Gasteiger, P., Leone, N., Musitsch, B., Ricca, F., Shchekotykhin, K.: Combining Answer Set Programming and domain heuristics for solving hard industrial problems (Application Paper). TPLP **16**(5-6), 653–669 (2016)
28. Dodaro, C., Leone, N., Nardi, B., Ricca, F.: Allotment problem in travel industry: A solution based on ASP. In: RR 2015. LNCS, vol. 9209, pp. 77–92. Springer (2015)
29. Eiter, T., Ianni, G., Krennwallner, T.: Answer set programming: A primer. In: Reasoning Web. LNCS, vol. 5689, pp. 40–110. Springer (2009)
30. Eiter, T., Leone, N., Saccà, D.: On the partial semantics for disjunctive deductive databases. Ann. Math. Artif. Intell. **19**(1-2), 59–96 (1997)
31. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming. AI Magazine **37**(3), 53–68 (2016)
32. Gaggl, S.A., Manthey, N., Ronca, A., Wallner, J.P., Woltran, S.: Improved answer-set programming encodings for abstract argumentation. TPLP **15**(4-5), 434–448 (2015)
33. Galindo, M.J.O., Ramírez, J.R.A., Carballido, J.L.: Logical weak completions of paraconsistent logics. J. Log. Comput. **18**(6), 913–940 (2008)
34. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Morgan & Claypool Publishers (2012)
35. Gebser, M., Leone, N., Maratea, M., Perri, S., Ricca, F., Schaub, T.: Evaluation techniques and systems for answer set programming: a survey. In: IJCAI'18. pp. 5450–5456 (2018)
36. Gebser, M., Maratea, M., Ricca, F.: The Design of the Sixth Answer Set Programming Competition. In: LPNMR'15. pp. 531–544 (2015)
37. Gebser, M., Maratea, M., Ricca, F.: What's hot in the answer set programming competition. In: AAAI. pp. 4327–4329. AAAI Press (2016)
38. Gebser, M., Maratea, M., Ricca, F.: The design of the seventh answer set programming competition. In: Balduccini, M., Janhunen, T. (eds.) LPNMR. LNCS, vol. 10377, pp. 3–9. Springer (2017)
39. Gebser, M., Maratea, M., Ricca, F.: The sixth answer set programming competition. Journal of Artif. Intell. Res. **60**, 41–95 (2017)
40. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. **9**(3/4), 365–386 (1991)
41. Giunchiglia, E., Leone, N., Maratea, M.: On the relation among answer set solvers. Ann. Math. Artif. Intell. **53**(1-4), 169–204 (2008)
42. Giunchiglia, E., Maratea, M.: On the Relation Between Answer Set and SAT Procedures (or, Between cmodels and smodels). In: ICLP. LNCS, vol. 3668, pp. 37–51 (2005)
43. Lierler, Y., Maratea, M., Ricca, F.: Systems, engineering environments, and competitions. AI Magazine **37**(3), 45–52 (2016)
44. Lifschitz, V., Turner, H.: Splitting a logic program. In: ICLP. pp. 23–37. MIT Press (1994)
45. Manna, M., Ricca, F., Terracina, G.: Taming primary key violations to query large inconsistent data via ASP. TPLP **15**(4-5), 696–710 (2015)

46. Maratea, M., Pulina, L., Ricca, F.: A multi-engine approach to answer-set programming. TPLP **14**(6), 841–868 (2014)
47. 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)
48. Pearce, D.: Equilibrium logic. Ann. Math. Artif. Intell. **47**(1-2), 3–41 (2006)
49. Pereira, L.M., Pinto, A.M.: Approved models for normal logic programs. In: LPAR. pp. 454–468 (2007)
50. Przymusinski, T.C.: Stable semantics for disjunctive programs. New Generation Comput. **9**(3/4), 401–424 (1991)
51. Sakama, C., Inoue, K.: Paraconsistent stable semantics for extended disjunctive programs. J. Log. Comput. **5**(3), 265–285 (1995)
52. Seipel, D.: Partial evidential stable models for disjunctive deductive databases. In: LPKR. LNCS, vol. 1471, pp. 66–84. Springer (1997)
53. van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. J. ACM **38**(3), 620–650 (1991)
54. You, J., Yuan, L.: A three-valued semantics for deductive databases and logic programs. J. Comput. Syst. Sci. **49**(2), 334–361 (1994)