TPLP **19** (5–6): 757–772, 2019. © Cambridge University Press 2019
doi:10.1017/S1471068419000176

Better Paracoherent Answer Sets with Less Resources

GIOVANNI AMENDOLA, CARMINE DODARO and FRANCESCO RICCA
University of Calabria, Rende, Italy
(*e-mail:* {amendola,dodaro,ricca}@mat.unical.it)

submitted 25 July 2019; accepted 31 July 2019

Abstract

Answer Set Programming (ASP) is a well-established formalism for logic programming. Problem solving in ASP requires to write an ASP program whose answers sets correspond to solutions. Albeit the non-existence of answer sets for some ASP programs can be considered as 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 programs, with the result of increasing the range of applications of ASP. State of the art implementations of paracoherent ASP adopt the semi-equilibrium semantics, but cannot be lifted straightforwardly to compute efficiently the (better) split semi-equilibrium semantics that discards undesirable semi-equilibrium models. In this paper an efficient evaluation technique for computing a split semi-equilibrium model is presented. An experiment on hard benchmarks shows that better paracoherent answer sets can be computed consuming less computational resources than existing methods.

KEYWORDS: Answer Set Programming, Paracoherent reasoning, Semi-equilibrium models

1 Introduction

In the past decades, key advances in Artificial Intelligence research were made thanks to studies in the field of Knowledge Representation and Reasoning (KRR) (van Harmelen et al. 2008). Among the established paradigms of KRR is Answer Set Programming (ASP) (Baral 2003; Brewka et al. 2011; Gebser et al. 2012), which is a well-known formalism for logic programming and non-monotonic reasoning. ASP is based on the stable model (or answer set) semantics (Gelfond and Lifschitz 1991), and features efficient implementations (Lierler et al. 2016; Gebser et al. 2016), such as CLASP (Gebser et al. 2015; Gebser et al. 2019), WASP (Alviano et al. 2015; Alviano et al. 2019), and DLV (Leone et al. 2006; Alviano et al. 2017). Problem solving in ASP requires to write an ASP program whose answers sets correspond to solutions (Lifschitz 1999), and then to compute these solutions resorting to an ASP solver. The availability of efficient implementations made possible the development of concrete applications, and as a matter of fact, ASP has been successfully applied to solve complex problems in Artificial Intelligence (Balduccini et al. 2001; Erdem et al. 2016; Gaggl et al. 2015; Dodaro et al. 2015); Bioinformatics (Campeotto et al. 2015); Databases (Arenas et al. 2003), and industrial applications (Dodaro et al. 2016).

The non-existence of answer sets for some ASP programs can be a modeling feature, but, as argued in (Amendola et al. 2016), 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* (Eiter et al. 2010). In particular, Eiter, Fink and Moura improved the paracoherent semantics of *semi-stable models* (Sakama and Inoue 1995) avoiding some anomalies with respect to basic modal logic properties by resorting to equilibrium logic (Pearce 2006). Thus, this paracoherent semantics is called *semi-equilibrium model (SEQ) semantics* (Eiter et al. 2010). More recently, (Amendola et al. 2016) noticed that, although the SEQ semantics has nice properties, it may select models that do not respect the 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, (Amendola et al. 2016) refine SEQ-models using splitting sequences (Lifschitz and Turner 1994), the major tool for modularity in modeling and evaluating answer set programs. In particular, the refined semantics, called *Split SEQ model semantics*, is able to discard model M_2.

The first efficient implementations of paracoherent semantics were proposed recently (Amendola et al. 2017; Amendola et al. 2018), 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 cannot be lifted straightforwardly to compute the refined semantics efficiently. Consequently, *no implementation of Split SEQ semantics has been available up to now.*

In this paper, we fill this lack presenting the first efficient strategy for computing a split SEQ model. In particular, we introduce an elegant program transformation, obtained by using weak constraints with levels (Buccafurri et al. 2000), that allows for computing a split semi-equilibrium model using a single call to a plain ASP solver, and prove a non-obvious correctness result. Notably, we exploited the modularity property of split semi equilibrium models to simplify the transformation and avoid the introduction of some rules and symbols that are needed in state of the art epistemic-transformation-based methods for computing SEQ models.

We have implemented the new approach and run an experiment to validate it empirically. Actually, no direct comparison with an alternative methods can be done, since ours is the first implementation of split SEQ models. Nonetheless, since split SEQ models are also SEQ models, we could compare it against existing implementations for semi-equilibrium models. As done previoulsy in the literature (Amendola et al. 2017; Amendola et al. 2018), we considered hard benchmarks from ASP competitions (Calimeri et al. 2016) modeling an application of paracoherent semantics to debugging (Cuteri et al. 2019). The experiment demonstrates that the new method outperforms state of the art methods for computing SEQ models consuming less computational resources, i.e., it uses less memory and terminates in less time.

The paper is structured as follows: preliminary notions on ASP and paracoherent answer sets are reported in Section 2; the description of strategies for computing split SEQ models is provided in Section 3; the empirical validation of our approach is presented in Section 4; related work is compared and discussed in Section 5; finally, we draw the conclusion in Section 6.

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

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

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]. The set of all atoms occurring in the program P is denoted by $At(P) = \bigcup_{r \in P} At(r)$.

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* (Gelfond and Lifschitz 1991) 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*.

In the following, we will also use *constraints*, which are of the form

$$\leftarrow b_1, ..., b_m, not\ c_1, ..., not\ c_n,$$

with $m, n \geq 0$, to be considered as a shorthand for a rule

$$\gamma \leftarrow b_1, ..., b_m, not\ c_1, ..., not\ c_n, not\ \gamma,$$

using a fresh atom γ that is not occurring elsewhere in the program. Note that ASP solvers do normally not create any auxiliary symbols for constraints.

Moreover, we recall a useful extension of the answer set semantics by the notion of *weak constraint* (Buccafurri et al. 2000). A weak constraint ω is of the form:

$$:\sim b_1, \ldots, \ b_m, \ not \ c_1, \ldots, \ not \ c_n. \ [w@l],$$

where w and l are nonnegative integers, representing a *weight* and a *level*, respectively. Let $\Pi = P \cup \Omega$, where P is a set of rules and Ω is a set of weak constraints. We call M an answer set of Π if it is an answer set of P. We denote by $\Omega(l)$ the set of all weak constraints at level l. For every answer set M of Π and any l, the *penalty* of M at level l, denoted by $Penalty_\Pi(M, l)$, is defined as $\sum_{\omega \in \Omega(l),\ M \models B(\omega)} w$. In case Π is clear from the context, we omit the subscript. For any two answer sets M and M' of Π, we say M is *dominated* by M' if there is l s.t. (i) $Penalty_\Pi(M', l) < Penalty_\Pi(M, l)$ and (ii) for all integers $k > l$, $Penalty_\Pi(M', k) = Penalty_\Pi(M, k)$. An answer set of Π is *optimal* if it is not dominated by another one of Π. We denote by $AS^O(\Pi)$ the set of all optimal answer sets of Π.

2.2 Semi-Equilibrium Models

Here, we introduce the paracoherent semantics of the *semi-equilibrium (SEQ) models* introduced in (Eiter et al. 2010). 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:

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

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

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 SEQ Models

A set $S \subseteq At(P)$ is a *splitting set* of P, if for every rule r in P such that $head(r) \cap S \neq \emptyset$ 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 models 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 (Amendola et al. 2016)), 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-equilibrium models. Finally, note that $M^{SCC}(P) \subseteq SEQ(P)$.

Example 1
Consider the program

$$P = \left\{ \begin{array}{lcl} b & \leftarrow & not\ a; \\ d & \leftarrow & b,\ not\ c; \\ c & \leftarrow & d \end{array} \right\}.$$

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 both SEQ models and split SEQ models as *paracoherent answer sets*.

2.4 Complexity Considerations

The complexity of various reasoning tasks with paracoherent answer sets has been analyzed in (Amendola et al. 2016). In the general case, checking whether an atom a is true in *some* paracoherent answer set (brave reasoning) is Σ_3^P-complete; whereas checking whether an atom a is true in *all* paracoherent answer sets (cautious reasoning) is Π_3^P-complete. However, computing a paracoherent answer set is feasible in $\mathrm{F}\Delta_3^P$, because it

is sufficient to find a paracoherent answer set that is cardinality minimal with respect to the gap.

3 On the Computation of Split SEQ Models

We start to note that an efficient computation of a split semi-equilibrium model requires a deep theoretical understanding of the paracoherent semantics. Indeed, a naive implementation of the split semi-equilibrium semantics considers each possible path that can be generated through the splitting sequence. Since each subprogram could have more than one answer set, one should explore an exponential number of paths. Note that, each path generated through the splitting sequence leads to obtain a paracoherent answer set of the last program (i.e., $P_n \cup \mathit{info}(M_{n-1})$), which is not necessarily a paracoherent answer set of the entire program P because it must also be gap-minimal.

Example 2
Consider the program

$$P = \left\{ \begin{array}{lcl} a & \leftarrow & \mathit{not}\ b; \\ b & \leftarrow & \mathit{not}\ a; \\ c & \leftarrow & a,\ \mathit{not}\ c \end{array} \right\}.$$

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 $\{b, Kb\}$. Hence, $\{a, Ka, Kc\}$ cannot be a paracoherent answer set of P, because it has a larger gap w.r.t. $\{b, Kb\}$. Indeed, $\mathscr{G}(\{a, Ka, Kc\}) = \{Kc\} \supset \mathscr{G}(\{b, Kb\}) = \emptyset$.

From a computational view point, a naive approach is very expensive, as one has to enumerate all possible paracoherent models obtainable from each path to make feasible a final phase of gap minimization.

Hence, we developed a clever strategy to compute a split SEQ model. Given a program P and a topological ordering $(C_1, \ldots, C_n)$ of $SCC(P)$, we construct a new program $\mathit{split}(P)$ which is the union of P^{HT} with the program $P_\gamma = \{\gamma a \leftarrow Ka, \mathit{not}\ a \mid a \in At(P)\}$, and the following set Ω of weak constraints. For each $i = 1, \ldots, n$, and for each atom $a \in C_i$, the weak constraint $:\sim \gamma a\ [1 : n - i]$ belongs to Ω. Then, we define

$$\mathit{split}(P) = P^{HT} \cup P_\gamma \bigcup_{i=1}^{n} \{:\sim \gamma a\ [1 : n - i] \mid a \in C_i\}.$$

We will show that an optimal answer set of $\mathit{split}(P)$ is a split SEQ model of P. Hence, in particular, it is also a SEQ model of P. First, we highlight a fundamental relation between the penalty of a model at a fixed level and the set of gap atoms of that model belonging to a strongly connected component.

Proposition 1
Let M be an optimal answer set of $\mathit{split}(P)$. Then, for $l = 0, ..., n - 1$,

$$\mathit{Penalty}(M, l) = |\mathscr{G}(M) \cap C_{n-l}|.$$

Proof Sketch
Since $\Omega(l) = \{:\sim \gamma a\ [1:l] \mid a \in C_{n-l}\}$, we obtain that

$$Penalty(M,l) = \sum_{\omega \in \Omega(l), M \models B(\omega)} w = \sum_{:\sim \gamma a \in \Omega(l), M \models \{\gamma a\}} 1 = \sum_{a \in C_{n-l}, \gamma a \in M} 1 = |\mathscr{G}(M) \cap C_{n-l}|.$$

□

Now we can prove our main result.

Theorem 1
Let P be a program and M be an optimal answer set of $split(P)$. Then, $M \setminus \{\gamma a \mid a \in At(P)\}$ is a split SEQ model of P.

Proof
We prove the claim by induction on the cardinality of $|SCC(P)|$.

In case of $|SCC(P)| = 1$, we have a unique strongly connected component, say C, of P. Hence, we have to consider the unique topological ordering (C). It leads to have the following set of weak constraints $\Omega = \Omega(1) = \{:\sim \gamma a\ [1:0] \mid a \in C\}$. Hence, M is an answer set of $P^{HT} \cup P_\gamma$ such that a minimum number of weak constraints in Ω is violated. This means that M is cardinality minimal with respect to the gap atoms. Therefore, it is also subset minimal with respect to the gap atoms, and so, $M' = M \setminus \{\gamma a \mid a \in At(P)\}$ is a semi-equilibrium model of P. As for $n = 1$, the split semi-equilibrium models of P coincide with the semi-equilibrium models of P, then M' is a split semi-equilibrium model of P.

Now, assume the claim holds in case of programs with $n-1$ strongly connnected components, and we want to prove that it also holds for programs with n strongly connected components. Let $SCC(P) = \{C_1, \ldots, C_n\}$, and let $(P_1, \ldots, P_n)$ be the corresponding stratification for P. Let $M_{n-1} = M \cap (C_{n-1} \cup \{Ka \mid a \in C_{n-1}\})$. By construction, $M_{n-1} \in AS^O((P^{HT} \setminus P_n^{HT}) \cup P_\gamma \cup (\Omega \setminus \Omega(n)))$. Hence, by inductive hypothesis, $M_{n-1} \setminus \{\gamma a \mid a \in At(P)\}$ is a split semi-equilibrium model of $P \setminus P_n$. Now, we have to prove that $M' = M \setminus \{\gamma a \mid a \in At(P)\} \in SEQ(P_n \cup info(M_{n-1}))$. Consider the program $\Pi = P_n^{HT} \cup info(M_{n-1}) \cup \Omega(n)$. First, (i) $M \in AS(P_n^{HT} \cup info(M_{n-1}))$, as $M \in AS(P^{HT})$. Second, (ii) M violates a minimum number of weak constraints in $\Omega(n)$. Indeed, by contradiction, there exists I violating a strictly less number of weak constraints in $\Omega(n)$ than M. So that, such a I dominates M with respect to $split(P)$, against the assumption that M is an optimal answer set of $split(P)$. Then, by (i) and (ii), it holds that $M \in AS^O(\Pi)$. Therefore, $M' \in SEQ(P_n \cup info(M_{n-1}))$. Finally, we claim that M' is maximal canonical among the interpretations over $At(P)$ satisfying conditions (1), (2) and (3) in Definition 3. Assume, by contradiction, that there is an interpretation I satisfying conditions (1), (2) and (3) in Definition 3 such that $\mathscr{G}(I) \subset \mathscr{G}(M)$. Hence, by Proposition 1, there is some nonnegative integer l such that (i) $Penalty(I,l) < Penalty(M,l)$ and (ii) for all integers $k > l$, $Penalty(I,k) = Penalty(M,k)$. Then, M is *dominated* by I. Thus, M is not an optimal answer set of $split(P)$, against the hypothesis. □

Note that the split semi-equilibrium model (hence the semi-equilibrium model) that such an algorithm will find, generally, is not cardinality minimal among the split semi-equilibrium models of the given program. This is coherent with complexity results, indeed computing optimal answer sets of ASP programs with weak constraints is known to be

a $F\Delta_3^P$ task for disjunctive programs (Buccafurri et al. 2000), and our technique can be implemented by an algorithm that runs in polynomial time (indeed, it consists of two polynomial tasks: the construction of the epistemic transformation and the computation of the SCCs).

Example 3
Consider, for instance, the following program

$$P = \left\{ \begin{array}{lcl} a & \leftarrow & not\ b; \\ b & \leftarrow & not\ a; \\ c & \leftarrow & b,\ not\ c; \\ d & \leftarrow & a,\ not\ c,\ not\ d; \\ e & \leftarrow & d \end{array} \right\}.$$

Hence, we have to consider the stratification of P given by $P_1 = \{a \leftarrow not\ b;\ b \leftarrow not\ a\}$; $P_2 = \{c \leftarrow b,\ not\ c\}$; $P_3 = \{d \leftarrow a,\ not\ c,\ not\ d\}$; and $P_4 = \{e \leftarrow d\}$. At the first step, we obtain $\{a, Ka\}$ and $\{b, Kb\}$ as SEQ models of P_1. At the second step, $\{a, Ka\}$ is the SEQ model of $P_2 \cup info(\{a, Ka\})$, and $\{b, Kb, Kc\}$ is the SEQ model of $P_2 \cup info(\{b, Kb\})$. At the third step, $\{a, Ka, Kd\}$ is the SEQ model of $P_3 \cup info(\{a, Ka\})$, and $\{b, Kb, Kc\}$ is the SEQ model of $P_3 \cup info(\{b, Kb, Kc\})$. At the fourth and final step, $\{a, Ka, Kd, Ke\}$ is the SEQ model of $P_4 \cup info(\{a, Ka, Kd\})$, and $\{b, Kb, Kc\}$ is the SEQ model of $P_4 \cup info(\{b, Kb, Kc\})$. Therefore, $\{a, Ka, Kd, Ke\}$ and $\{b, Kb, Kc\}$ are the split SEQ models of P. However, $\{a, Ka, Kd, Ke\}$ is preferred to $\{b, Kb, Kc\}$. Indeed, $\{b, Kb, Kc\}$ violates the weak constraint $:\sim \gamma c$, that is at a lower level than $:\sim \gamma d$ and $:\sim \gamma e$, that are violated by $\{a, Ka, Kd, Ke\}$.

Moreover, the split SEQ semantics allows to make a fundamental simplification of symbols in the HT-epistemic transformation of the program. Indeed, given a program P and a stratification for P in subprograms $(P_1, \ldots, P_n)$, whenever P_1, ..., P_k, with $k < n$, are coherent programs, we have no need to compute the HT-epistemic transformation of P_1, ..., P_k, but we can directly compute the answer sets of $P_1 \cup \cdots \cup P_k$. So that, if $M \in AS(P_1 \cup \cdots \cup P_k)$, then $M \cup \{Ka \mid a \in M\}$ is a paracoherent answer set of $P_1 \cup \cdots \cup P_k$.

Theorem 2
Let P be a program, $(P_1, \ldots, P_n)$ be a stratification for P, and k be the maximal number so that P_j is coherent, for each $j = 1, ..., k$. Then,

$$M^{SCC}(P) = \{I^K \cup J \mid I \in AS(P_{coh}) \wedge J \in M^{SCC}(P^I_{inc})\},$$

where $I^K = I \cup \{Ka \mid a \in I\}$; $P_{coh} = P_1 \cup \cdots \cup P_k$; $P_{inc} = P_{k+1} \cup \cdots \cup P_n$; and P^I_{inc} comes from P_{inc} by removing each rule r s.t. $B^-(r) \cap I \neq \emptyset$, and each atom in $At(P_{inc}) \cap I$.

Finally, note that to check if a program is coherent, is a well-known Σ_2^P-complete problem. Hence, in the implementation we need to consider a *sufficient* condition to detect coherent programs in polynomial time. It is known that, if a program has no cycle in the dependency graph having an odd number of negated arcs, then it is coherent.

Proposition 2
Given a program P, detecting a cycle in the dependency graph of P having an odd number of negated arcs, can be done in linear time with respect to $|E_P|$.

Proof Sketch
Let $DG(P)$ be the dependency graph of P. We consider a directed graph G' such that for each positive arc in $DG(P)$, namely (a, b), we introduce a fresh node, namely ab, and replace (a, b), by two edges $\{a, ab\}$ and $\{ab, b\}$. Hence, if $DG(P)$ contains a cycle having an odd number of negated arcs then G' contains a cycle of odd length. The claim holds by the fact the a directed graph does not contain a directed cycle of odd length if, and only if, it is bipartite when treated as an undirected graph, i.e., it can be colored with two colors. This check can be done in linear time with respect to the number of arcs in the input graph (Kleinberg and Tardos 2006). □

We conclude this section observing that Theorem 1 and Theorem 2 hold (without modifications) also if one considers the *extended externally supported program* P^{es}, introduced in (Amendola et al. 2018) to characterize semi-equilibrium models, in place of the HT-epistemic transformation.

4 Experiments

In this section we present the results of an experimental analysis conducted to analyze the performance of the new strategy for computing a split semi-equilibrium model presented in the previous section.

4.1 Implementation

To compute a split semi-equilibrium model we have implemented in a rewriter tool a program transformation that takes as input an ASP program P and produces as output an optimized version of $split(P)$. In particular, the rewriter implements the efficient epistemic transformation P^{es} of (Amendola et al. 2018), which is known to be much more efficient than the classic HT-epistemic transformation, and implements the Tarjan algorithm (see e.g., (Kleinberg and Tardos 2006)) to compute a topological order of the $SCC(P)$ and build the weak constraints Ω. During the rewriting process, the components are also subject to a (modified) two-colorability check (see Proposition 2) applied following the topological order to optimize the output as indicated in Theorem 2. Thus, a split semi-equilibrium model is computed by evaluating the optimized $split(P)$ with the ASP solver WASP (Alviano et al. 2015; Alviano and Dodaro 2016).

4.2 Experimental Setting

A practical approach for the computation of a split semi-equilibrium model has been presented for the first time in this paper. Therefore, it is not possible to compare the performance of our implementation with alternative approaches for computing split semi-equilibrium models. Since all split semi-equilibrium models are semi-equilibrium models, we focus on the task of computing a semi-equilibrium model, and compare our implementation with the state of the art ones presented in (Amendola et al. 2018), namely: SPLIT, MIN, and WEAK. Note that, the labels correspond to the algorithm names used in (Amendola et al. 2018), where SPLIT is in no way related to the split semi equilibrium semantics; rather, the name SPLIT was chosen to remind the behavior of the algorithm

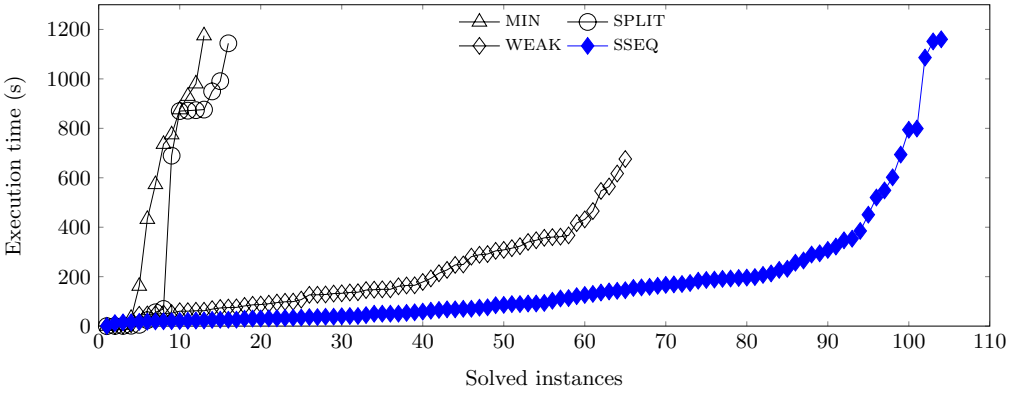


Fig. 1. Comparison of the best performing algorithms.

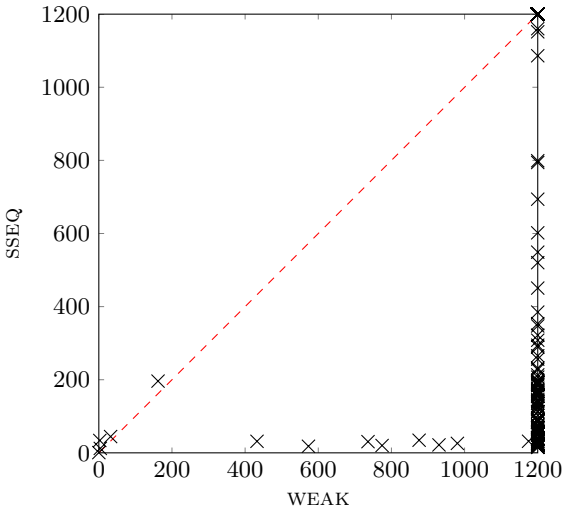
that "splits" the set of gap atoms of a candidate solution to perform gap minimization (for details see (Amendola et al. 2018)). In the following, our implementation is labeled SSEQ, since its distinguishing feature is to compute a split semi equilibrium model.

In order to perform a fair comparison, in this experiment we used the same version of the WASP solver, and the same experimental settings described in (Amendola et al. 2018). In particular, we considered all the incoherent instances from the latest ASP Competition (Gebser et al. 2017) that feature neither *aggregates*, nor *choice rules*, nor *weak constraints*, since such features are not supported by the paracoherent semantics (Amendola et al. 2016). This benchmark setting is of particular interest for paracoherent reasoning since it consists of debugging hard ASP programs, one of the main motivations of paracoherent ASP. In particular, the problem to be solved is the computation of an explanation for the non-existence of answer sets. Execution times and memory usage were limited to 1200 seconds and 8 GB, respectively.

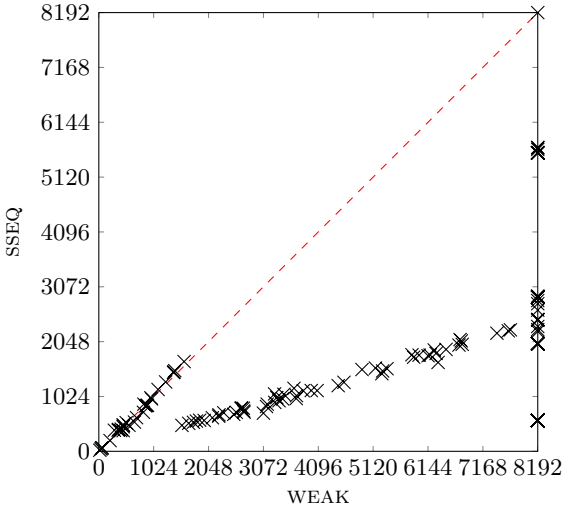
4.3 Results

The results of the experiment show that the new technique is better than state-of-the-art-approaches. In the vast majority of considered instances the improvements are significant, as seen from the cactus plots in Figure 1. In more detail, SSEQ solves overall 104 instances, whereas the performance achieved by MIN, SPLIT, and WEAK is considerably worse, solving 13, 16, and 65 instances in the allotted time, respectively.

Figures 2(a) and 2(b) show instance-by-instance comparisons for the two best-performing algorithms, i.e. SSEQ and WEAK. We recall that each plotted point represent an instance, and a point (x, y) is plotted if the two systems take x and y execution time (resp. memory usage) for evaluating the instance. Therefore, points below the diagonals represent instances where the system reported on the x-axis was slower (resp. uses more memory) than the system reported on the y-axis. The graphs confirm the better performance of SSEQ. Indeed, in Figure 2(a), only few instances are on the left of the diagonals, meaning that are only few instances where SSEQ is slower than WEAK. Concerning the memory usage, Figure 2(b) clearly shows SSEQ uses consistently less memory than WEAK. We also mention that SSEQ and WEAK exceed the allotted memory limited in 1 and 83 instances, respectively. It is worth reporting that disabling the optimization



(a) Runtimes.



(b) Memory usage.

Fig. 2. Instance-wise comparison between WEAK and SSEQ.

of Theorem 2, which is specific of split SEQ semantics, the resulting method could only solve 6 instances, and expectedly the performance w.r.t. memory consumption was also poorer, i.e., it exceeded 51 times the memory limits.

5 Related Work

Semantics for non-monotonic logic programs (Przymusinski 1991; van Gelder et al. 1991; You and Yuan 1994; Sakama and Inoue 1995; Eiter et al. 1997; Seipel 1997; Balduccini and Gelfond 2003; Pereira and Pinto 2007; Alcântara et al. 2005; Galindo et al. 2008; Amendola et al. 2016; Costantini and Formisano 2016) that

relax the definition of answer set to overcome the absence of answer sets can be considered in broader terms paracoherent semantics. Nonetheless, the first approach to the problem of handling inconsistency in ASP programs is the semi-stable semantics by (Sakama and Inoue 1995). Later (Eiter et al. 2010) identified some anomalies of semi-stable semantics with respect to some epistemic properties, and proposed the semi-equilibrium semantics. Notably, (Eiter et al. 2010) also introduced the term paracoherent for the semantics that provide a remedy to the absence of answer sets due to cyclic negation. In (Amendola et al. 2016) it was demonstrated that semi-equilibrium semantics features a number of highly desirable theoretical properties for a knowledge representation language, that are not all fulfilled by previous proposals: (i) every consistent answer set of a program corresponds to a paracoherent answer set (*answer set coverage*); (ii) if a program has some (consistent) answer set, then its paracoherent answer sets correspond to answer sets (*congruence*); (iii) if a program has a classical model, then it has a paracoherent answer set (*classical coherence*); (iv) a minimal set of atoms should be undefined (*minimal undefinedness*); (v) every true atom must be derived from the program (*justifiability*). The first two properties ensure that the notions of answer sets and paracoherent answer sets should coincide for coherent programs; the third states that paracoherent answer set should exist whenever the programs admits a (classical) model; the last two state that the number of undefined atoms should be minimized, and every true atom should be derived from the program, respectively. 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* (Lifschitz and Turner 1994) 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 (Eiter et al. 2009) is based on this property) and is a principle underlying the architectures of ASP systems (Lierler et al. 2016). The split semi-equilibrium semantics (Amendola et al. 2016) 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 (Amendola et al. 2017) 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 (Amendola et al. 2018) 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 (Amendola et al. 2017) nor (Amendola et al. 2018) support SSEQ semantics that is the focus of this paper.

For the sake of completeness, we mention that the algorithms used for computing paracoherent answer sets are strictly related to the computation of minimal models of propositional theories (Niemelä 1996; Bry and Yahya 2000; Koshimura et al. 2009; Janota and Marques-Silva 2016; Angiulli et al. 2014); the reader si referred to (Amendola et al. 2017) for a detailed discussion.

6 Conclusion and Future Work

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 (Eiter et al. 2010). 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 (Amendola et al. 2017; Amendola et al. 2018). State of the art solutions supported the semi-equilibrium semantics but cannot compute the (better) split semi-equilibrium semantics; notably, existing evaluation techniques cannot be adapted straightforwardly to accomplish this task. We remark that, as mentioned previously, split semi-equilibrium models have to be considered better in the sense that they are models that respect the modular structure of the program (as observed in (Amendola et al. 2016)), and, thus, they better fit the intentions of a programmer which usually exploits modularity to produce programs (e.g., by applying the guess and check methodology).

In this paper we presented a novel optimized program transformation that allows for computing a split semi-equilibrium model using a plain ASP solver. The transformation is elegant and independent from the epistemic transformation used to define semi-equilibrium models. Moreover, the modularity property of split semi equilibrium models allowed us to devise an optimization that further simplifies the transformed program and improves performance. We implemented the optimized transformation and run an experiment comparing it against existing implementations for semi-equilibrium models. Our implementation outperformed the state of the art methods in terms of both memory consumption and solving times, and it was able to solve 160% more instances than the best alternative solution using the same ASP solver. In conclusion, the paper shows how *better semi equilibrium models can be computed also more efficiently.*

The availability of efficient methods for computing one paracoherent answer set makes reasonable to start approaching more complex reasoning problems connected with the enumeration of paracoherent answer sets. Thus, as far as future work is concerned, we plan to investigate the extension of our techniques to the implementation of cautious and brave reasoning, e.g., on the lines of (Alviano 2018). Notably, this will not be a straightforward porting.

Finally, we mention that an interesting feature work is to investigate how to extend paracoherent rewriting techniques to non-ground ASP programs. Actually, our implementation supports non-ground ASP programs by simply disabling grounding simplifications and then using the resulting instantiation as input for the rewriting techniques applied later on. However, this may cause a deterioration of the performance since grounding simplifications have been shown to be useful for improving the performance of ASP solvers. Therefore, we plan to investigate if more sophisticated rewriting techniques can be directly applied to non-ground ASP programs.

References

Alcântara, J., Damásio, C. V., and Pereira, L. M. 2005. An encompassing framework for paraconsistent logic programs. *Journal of Applied Logic 3,* 1, 67–95.

Alviano, M. 2018. Query answering in propositional circumscription. In *Proceedings of the International Conference on Artificial Intelligence (IJCAI).* ijcai.org, 1669–1675.

Alviano, M., Amendola, G., Dodaro, C., Leone, N., Maratea, M., and Ricca, F. 2019. Evaluation of disjunctive programs in WASP. In *Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*. Lecture Notes in Computer Science, vol. 11481. Springer, 241–255.

Alviano, M., Calimeri, F., Dodaro, C., Fuscà, D., Leone, N., Perri, S., Ricca, F., Veltri, P., and Zangari, J. 2017. The ASP system DLV2. In *Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*. Lecture Notes in Computer Science, vol. 10377. Springer, 215–221.

Alviano, M. and Dodaro, C. 2016. Anytime answer set optimization via unsatisfiable core shrinking. *Theory and Practice of Logic Programming 16,* 5-6, 533–551.

Alviano, M., Dodaro, C., Leone, N., and Ricca, F. 2015. Advances in WASP. In *Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*. Lecture Notes in Computer Science. 40–54.

Amendola, G., Dodaro, C., Faber, W., Leone, N., and Ricca, F. 2017. On the computation of paracoherent answer sets. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 1034–1040.

Amendola, G., Dodaro, C., Faber, W., and Ricca, F. 2018. Externally supported models for efficient computation of paracoherent answer sets. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 1720–1727.

Amendola, G., Eiter, T., Fink, M., Leone, N., and Moura, J. 2016. Semi-equilibrium models for paracoherent answer set programs. *Artificial Intelligence 234*, 219–271.

Angiulli, F., Ben-Eliyahu, R., Fassetti, F., and Palopoli, L. 2014. On the tractability of minimal model computation for some CNF theories. *Artificial Intelligence 210*, 56–77.

Arenas, M., Bertossi, L. E., and Chomicki, J. 2003. Answer sets for consistent query answering in inconsistent databases. *Theory and Practice of Logic Programming 3,* 4-5, 393–424.

Balduccini, M. and Gelfond, M. 2003. Logic programs with consistency-restoring rules. In *Proceedings of the International Symposium on Logical Formalization of Commonsense Reasoning, AAAI Spring Symposium Series*. Vol. 102. 9–18.

Balduccini, M., Gelfond, M., Watson, R., and Nogueira, M. 2001. The usa-advisor: A case study in answer set planning. In *Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*. Lecture Notes in Computer Science, vol. 2173. Springer, 439–442.

Baral, C. 2003. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press, New York, NY, USA.

Brewka, G., Eiter, T., and Truszczynski, M. 2011. Answer set programming at a glance. *Communications of the ACM 54,* 12, 92–103.

Bry, F. and Yahya, A. H. 2000. Positive unit hyperresolution tableaux and their application to minimal model generation. *Journal of Automated Reasoning 25,* 1, 35–82.

Buccafurri, F., Leone, N., and Rullo, P. 2000. Enhancing disjunctive datalog by constraints. *IEEE Transactions on Knowledge and Data Engineering 12,* 5, 845–860.

Calimeri, F., Gebser, M., Maratea, M., and Ricca, F. 2016. Design and results of the fifth answer set programming competition. *Artificial Intelligence 231*, 151–181.

Campeotto, F., Dovier, A., and Pontelli, E. 2015. A declarative concurrent system for protein structure prediction on GPU. *Journal of Experimental & Theoretical Artificial Intelligence 27,* 5, 503–541.

Costantini, S. and Formisano, A. 2016. Query answering in resource-based answer set semantics. *Theory and Practice of Logic Programming 16,* 5-6, 619–635.

Cuteri, B., Dodaro, C., and Ricca, F. 2019. Debugging of answer set programs using paracoherent reasoning. In *Proceedings of the Italian Conference on Computational Logic (CILC)*. CEUR Workshop Proceedings, vol. 2396. CEUR-WS.org, 289–299.

Dodaro, C., Gasteiger, P., Leone, N., Musitsch, B., Ricca, F., and Shchekotykhin, K. 2016. Combining Answer Set Programming and domain heuristics for solving hard industrial problems (Application Paper). *Theory and Practice of Logic Programming 16,* 5-6, 653–669.

Dodaro, C., Leone, N., Nardi, B., and Ricca, F. 2015. Allotment problem in travel industry: A solution based on ASP. In *Proceedings of the International Conference on Web Reasoning and Rule Systems (RR)*. Lecture Notes in Computer Science, vol. 9209. Springer, 77–92.

Eiter, T., Fink, M., and Moura, J. 2010. Paracoherent answer set programming. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Eiter, T., Ianni, G., and Krennwallner, T. 2009. Answer set programming: A primer. In *Proceedings of the Reasoning Web International Summer School*. Lecture Notes in Computer Science, vol. 5689. Springer, 40–110.

Eiter, T., Leone, N., and Saccà, D. 1997. On the partial semantics for disjunctive deductive databases. *Annals of Mathematics and Artificial Intelligence 19,* 1-2, 59–96.

Erdem, E., Gelfond, M., and Leone, N. 2016. Applications of answer set programming. *AI Magazine 37,* 3, 53–68.

Gaggl, S. A., Manthey, N., Ronca, A., Wallner, J. P., and Woltran, S. 2015. Improved answer-set programming encodings for abstract argumentation. *Theory and Practice of Logic Programming 15,* 4-5, 434–448.

Galindo, M. J. O., Ramírez, J. R. A., and Carballido, J. L. 2008. Logical weak completions of paraconsistent logics. *Journal of Logic and Computation 18,* 6, 913–940.

Gebser, M., Kaminski, R., Kaufmann, B., Romero, J., and Schaub, T. 2015. Progress in clasp series 3. In *Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*. Lecture Notes in Computer Science, vol. 9345. Springer, 368–383.

Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T. 2012. *Answer Set Solving in Practice*. Morgan & Claypool Publishers.

Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming 19,* 1, 27–82.

Gebser, M., Maratea, M., and Ricca, F. 2016. What's hot in the answer set programming competition. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 4327–4329.

Gebser, M., Maratea, M., and Ricca, F. 2017. The sixth answer set programming competition. *Journal of Artificial Intelligence Research 60*, 41–95.

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9,* 3/4, 365–386.

Janota, M. and Marques-Silva, J. 2016. On the query complexity of selecting minimal sets for monotone predicates. *Artificial Intelligence 233*, 73–83.

Kleinberg, J. M. and Tardos, É. 2006. *Algorithm design*. Addison-Wesley.

Koshimura, M., Nabeshima, H., Fujita, H., and Hasegawa, R. 2009. Minimal model generation with respect to an atom set. In *Proceedings of the International Workshop on First-Order Theorem Proving (FTP)*. CEUR Workshop Proceedings, vol. 556. CEUR-WS.org, 49–59.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., and Scarcello, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic 7,* 3, 499–562.

Lierler, Y., Maratea, M., and Ricca, F. 2016. Systems, engineering environments, and competitions. *AI Magazine 37,* 3, 45–52.

Lifschitz, V. 1999. Answer set planning. In *Proceedings of the International Conference on Logic Programming (ICLP)*. MIT Press, 23–37.

Lifschitz, V. and Turner, H. 1994. Splitting a logic program. In *Proceedings of the International Conference on Logic Programming (ICLP)*. MIT Press, 23–37.

Niemelä, I. 1996. A tableau calculus for minimal model reasoning. In *Proceedings of the International Workshop on Theorem Proving with Analytic Tableaux and Related Methods (TABLEAUX)*. 278–294.

Pearce, D. 2006. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence 47,* 1-2, 3–41.

Pereira, L. M. and Pinto, A. M. 2007. Approved models for normal logic programs. In *Proceedings of the International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)*. Lecture Notes in Computer Science, vol. 4790. 454–468.

Przymusinski, T. C. 1991. Stable semantics for disjunctive programs. *New Generation Computing 9,* 3/4, 401–424.

Sakama, C. and Inoue, K. 1995. Paraconsistent stable semantics for extended disjunctive programs. *Journal of Logic and Computation 5,* 3, 265–285.

Seipel, D. 1997. Partial evidential stable models for disjunctive deductive databases. In *Proceedings of the International Workshop on Logic Programming and Knowledge Representation (LPKR)*. Lecture Notes in Computer Science, vol. 1471. Springer, 66–84.

van Gelder, A., Ross, K. A., and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM 38,* 3, 620–650.

van Harmelen, F., Lifschitz, V., and Porter, B. W., Eds. 2008. *Handbook of Knowledge Representation*. Foundations of Artificial Intelligence, vol. 3. Elsevier.

You, J. and Yuan, L. 1994. A three-valued semantics for deductive databases and logic programs. *Journal of Computer and System Sciences 49,* 2, 334–361.