Externally Supported Models for Efficient Computation of Paracoherent Answer Sets

Giovanni Amendola[1] and **Carmine Dodaro**[2] and **Wolfgang Faber**[3] and **Francesco Ricca**[1]
[1]DEMACS, University of Calabria, Italy
[2]DIBRIS, University of Genova, Italy
[3]University of Huddersfield, UK
{amendola,ricca}@mat.unical.it, dodaro@dibris.unige.it, wf@wfaber.com

Abstract

Answer Set Programming (ASP) is a well-established formalism for nonmonotonic reasoning. While incoherence, the non-existence of answer sets for some programs, is an important feature of ASP, it has frequently been criticised and indeed has some disadvantages, especially for query answering. Paracoherent semantics have been suggested as a remedy, which extend the classical notion of answer sets to draw meaningful conclusions also from incoherent programs. In this paper we present an alternative characterization of the two major paracoherent semantics in terms of (extended) externally supported models. This definition uses a transformation of ASP programs that is more parsimonious than the classic epistemic transformation used in recent implementations. A performance comparison carried out on benchmarks from ASP competitions shows that the usage of the new transformation brings about performance improvements that are independent of the underlying algorithms.

1 Introduction

Knowledge Representation and Reasoning are a core topic in Artificial Intelligence and in the past decades, major progress has been achieved. In the subarea of nonmonotonic reasoning, Answer Set Programming (ASP) has become the primarily used formalism (cf. (Brewka, Eiter, and Truszczynski 2011; Gebser et al. 2012)). ASP offers a declarative language and allows for solving difficult, usually NP-hard, problems by encoding them as a logic program and computing its answer sets, which encode the problem solutions. The availability of efficient solvers has leveraged a large variety of applications (Gaggl et al. 2015; Manna, Ricca, and Terracina 2015; Amendola et al. 2016a; Dodaro et al. 2016; Amendola et al. 2016c), including industrial ones (Grasso et al. 2011).

One important, but in some circumstances peculiar, feature is that some logic programs have no answer sets. While this is sometimes desired for encoding problems that admit no solutions, it is sometimes perceived as detrimental, especially when dealing with query answering. Addressing this issue, paracoherent semantics based on answer sets have been proposed to draw meaningful conclusions also from incoherent programs (Amendola et al. 2016b). The term paracoherent has been chosen to highlight both similarities and differences to paraconsistent semantics: their goal is similar, but the latter addresses classical logical contradictions, while the former addresses contradictions due to unstratified ("cyclic") negation.

Practical applications of these paracoherent semantics hinge on the availability of efficient algorithms and implementations. There is a vast potential of applications, the most immediate ones being debugging of ASP and incoherence-tolerant query answering. But also applications in diagnosis, planning, and reasoning about actions are conceivable (Amendola et al. 2016b).

So far, we are aware of only one very recent attempt on providing efficient reasoning support for paracoherent semantics. The work presented in (Amendola et al. 2017) relies on the epistemic transformation of ASP programs underlying the theoretical foundations of the semantics (Amendola et al. 2016b), and constructs a few algorithms around it that build upon existing ASP solvers. The fact that this method relies on the epistemic transformation introduces considerable overhead, though. Indeed, it creates so many auxiliary atoms and additional rules that the system described in (Amendola et al. 2017) often does not terminate or even runs out of memory.

In this paper we present an alternative characterization of paracoherent answer sets in terms of (extended) externally supported models. The definition is based on a new transformation of the program that is more parsimonious than the classical one in terms of the number of additional atoms and the number of new rules. This transformation can be used in place of the epistemic transformation in existing solving algorithms without requiring any other modification. An empirical performance comparison on benchmarks from ASP competitions shows that significant performance improvements can be obtained by using the new transformation, and the advantages are independent of the used algorithms and underlying ASP solvers.

2 Preliminaries

We start with recalling the basic notions of answer set semantics, and then present the semi-stable and semi-equilibrium paracoherent semantics.

We concentrate on programs over a propositional signature Σ. A *rule r* is of the form

$$a_1 \vee \ldots \vee a_l \leftarrow b_1, \ldots, b_m, not\ c_1, \ldots, not\ c_n \quad (1)$$

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 $B(r) = B^+(r) \cup B^-(r)$. If $B(r) = \emptyset$, we then omit $\leftarrow$; and if $B^-(r) = \emptyset$, then r is *positive*. A *program* P is a finite set of rules. P is called *positive* if each $r \in P$ is positive.

Any set $I \subseteq \Sigma$ is an *interpretation*; it is a *model* of a program P (denoted $I \models P$) iff for each rule $r \in P$, $I \cap H(r) \neq \emptyset$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$ (denoted $I \models r$). A model M of P is *minimal* iff no model $M' \subset M$ of P exists. Given an interpretation I, we denote by P^I the well-known *Gelfond-Lifschitz reduct* (Gelfond and Lifschitz 1991) of P w.r.t. I, that is 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 $B^-(r) \cap I = \emptyset$. A model M of P is called *answer set (or stable model)* of P, if M is a minimal model of P^M. We denote by $AS(P)$ the set of all answer sets of P. We say that P is *consistent*, if it admits some model, otherwise it is *inconsistent*; whereas it is *coherent*, if it admits some answer set, otherwise, it is *incoherent*.

In the following, we will also use *choice rules* (Simons, Niemelä, and Soininen 2002) of the form $\{a\}$, where $a \in \Sigma$. A choice rule $\{a\}$ can be viewed as a syntactic shortcut for for the rule $a \vee a_F$, where a_F is a fresh new atom not appearing elsewhere in the program, meaning that the atom a can be chosen as true. Note that ASP solvers do normally not create any auxiliary symbols for choice rules.

Next, we introduce two paracoherent semantics. The first one is known as *semi-stable model semantics* and was introduced by (Sakama and Inoue 1995). We 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 semi-stable models of a program P are obtained from its *epistemic κ-transformation*.

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

$$\lambda_{r,1} \vee \ldots \vee \lambda_{r,l} \vee Kc_1 \vee \ldots \vee Kc_n \leftarrow b_1, \ldots, b_m; \quad (2)$$
$$a_i \leftarrow \lambda_{r,i}; \quad (3)$$
$$\leftarrow \lambda_{r,i}, c_j; \quad (4)$$
$$\lambda_{r,i} \leftarrow a_i, \lambda_{r,k}; \quad (5)$$

for $1 \leq i, k \leq l$ and $1 \leq j \leq n$, where the $\lambda_{r,i}$ are fresh atoms.

Given an interpretation I^κ over $\Sigma' \supseteq \Sigma^\kappa$, let $\mathscr{G}(I^\kappa) = \{Ka \in I^\kappa \mid a \notin I^\kappa\}$ denote the atoms believed true but not assigned true, also referred to as the gap of I^κ. Given a set $\mathscr{F}$ of interpretations over Σ', an interpretation $I^\kappa \in \mathscr{F}$ is *maximal canonical in* $\mathscr{F}$, if no $J^\kappa \in \mathscr{F}$ exists such that $\mathscr{G}(I^\kappa) \supset \mathscr{G}(J^\kappa)$. By $mc(\mathscr{F})$ we denote the set of maximal canonical interpretations in $\mathscr{F}$. Semi-stable models are then defined as maximal canonical interpretations among the answer sets of P^κ, and the set of all semi-stable models of P is denoted by $SST(P)$, i.e., $SST(P) = \{S \cap \Sigma^\kappa \mid S \in mc(AS(P^\kappa))\}$.

The second one is called *semi-equilibrium model semantics* and was introduced by (Amendola et al. 2016b) to amend anomalies in semi-stable model semantics. Semi-equilibrium models may be computed as maximal canonical answer sets of an extension of the epistemic κ-transformation.

Definition 2 (Epistemic HT-transformation P^{HT})**.** Let P be a program over Σ. Then its epistemic HT-transformation P^{HT} is defined as the union of P^κ with the set of rules:

$$Ka \leftarrow a, \quad (6)$$
$$Ka_1 \vee \ldots \vee Ka_l \vee Kc_1 \vee \ldots \vee Kc_n \leftarrow Kb_1, \ldots, Kb_m, \quad (7)$$

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

Then, the set of all semi-equilibrium models is given by $\{M \cap \Sigma^\kappa \mid M \in mc(AS(P^{HT}))\}$ and is denoted by $SEQ(P)$. In the following, we refer to semi-stable models or semi-equilibrium models as *paracoherent answer sets*.

3 Minimal Externally Supported Models

In this section, we introduce an alternative view on the previous paracoherent answer set semantics, that will have a strong impact on the computation of paracoherent answer sets. To this end, we will focus our attention on the concept of *supported atom*, and we will show that both semi-stable and semi-equilibrium semantics can be expressed in terms of *external* supports.

Let P be a program over Σ. We consider an extended signature $\Sigma^s = \Sigma \cup \{sa : a \in \Sigma\}$, where each sa is a new atom, called *support atom*. For each rule $r \in P$ of the form (1), we build a new rule r^s as follows:

$$a_1 \vee \ldots \vee a_l \leftarrow \quad b_1, \ldots, b_m, not\ c_1, \ldots, not\ c_n, \ not\ sc_1, \ldots, not\ sc_n; \quad (8)$$

Moreover, for each support atom sc, corresponding to a negated body atom c, we build the choice rule

$$\{sc\}. \quad (9)$$

Intuitively, these two kinds of rules mean that we consider each possible external support for each negated atom of the original program. Note that, as the support atoms in rule (8) appear only in the negated body (under the default negation), if no support atom is chosen from the set of rules of the form (9), then we obtain the original rule. We refer to the resulting program as the *externally supported program* with respect to P, denoted by P^s.

Finally, we also consider a program P^{es} obtained by extending P^s with the *support distribution rules*, defined as follows. For each rule $r \in P$ of the form (1), we consider a new rule r^d as

$$sa_1 \vee \ldots \vee sa_l \vee sc_1 \vee \ldots \vee sc_n \leftarrow \ sb_1, \ldots, sb_m, \ not\ a_1, \ldots, not\ a_l, \ not\ c_1, \ldots, not\ c_n; \quad (10)$$

We call $P^{es} = P^s \cup \{r^d : r \in P\}$ the *extended externally supported program* with respect to P.

Intuitively, the supported distribution rules allow to give a support to an atom of the original program starting from

an external support atom, only if that atom has not a support in the original program. E.g., consider the trivial program $P_1 = \{a\}$, stating that a has a support in P_1. The extended externally supported program is $P_1^{es} = \{a;\ sa \leftarrow not\ a\}$. Since a has a support in P_1^{es}, there is no reason to give an external support to a, so that sa is not derived.

Similarly to the epistemic κ-transformation and the epistemic HT-transformation, the existence of a classical model of the original program implies the existence of an answer set in the (extended) externally supported program.

Theorem 1. Let P be a consistent program. Then, P^s and P^{es} are coherent programs.

Given an interpretation I over Σ^s and an interpretation J over Σ^κ, we denote by $s(I)$ the set $\{sa : sa \in I\}$ of all support atoms in I, and by $\kappa(J)$ the set $\{Ka : Ka \in J\}$ of all believed true atoms in J, respectively.

Definition 3 (Minimal Externally Supported Models). Let P be a consistent program, and let M be an answer set of P^s [resp., P^{es}]. We say that M is a *minimal externally supported model* [resp., *minimal extended externally supported model*] of P, if there is no answer set M' of P^s [resp., P^{es}] s.t. $s(M') \subset s(M)$. We denote by $MES(P)$ [resp., $MEES(P)$] the set of all minimal externally supported models [resp., minimal extended externally supported models] of P.

Theorem 2. Let P be a coherent program. Then, $AS(P) = MES(P) = MEES(P)$.

Proof. Let $A \in AS(P)$ be an answer set of P, since $s(A) = \emptyset$ it holds that $A \models r$ for all $r \in P^s$ (resp. $r \in P^{es}$) (note that any additional rule of the form (9) and (10) is trivially satisfied since all support atoms are false in A), thus $A \in AS(P^s)$ (resp. $A \in AS(P^{es})$). Consequently (by Definition 3) we have that $A \in MES(P)$ (resp. $A \in MEES(P)$), i.e. $AS(P) \subseteq MES(P)$ (resp. $AS(P) \subseteq MEES(P)$). On the other hand, since for all $A \in AS(P)$ it holds that $s(A) = \emptyset$, there is no $A' \in AS(P^s)$ (resp. $A' \in AS(P^{es})$) such that $A' \notin AS(P)$ and $s(M') \subset s(M)$, i.e. $MES(P) \subseteq AS(P)$ (resp. $MEES(P) \subseteq AS(P)$). □

However, in general, the three semantics are very different, as shown by the following example.

Example 1. Consider the following incoherent program $P = \{a \leftarrow not\ b;\ b \leftarrow a, not\ c;\ c \leftarrow b;\ d \leftarrow c\}$. The externally supported program with respect to P is $P^s = \{a \leftarrow not\ b, not\ sb;\ b \leftarrow a, not\ c, not\ sc;\ c \leftarrow b;\ d \leftarrow c; \{sb\};\ \{sc\}\}$. The answer sets of P^s are $M_1 = \{sb\}$, $M_2 = \{a, sc\}$ and $M_3 = \{sb, sc\}$. However, M_3 is not a minimal externally supported model of P as, for instance, $s(M_3) = M_3$ strictly contains $s(M_1) = M_1$. Therefore, $MES(P) = \{M_1, M_2\}$. Instead, the extended externally supported program with respect to P is $P^{es} = P^s \cup \{sa \vee sb \leftarrow not\ a, not\ b;\ sb \vee sc \leftarrow sa, not\ b, not\ c;\ sc \leftarrow sb, not\ c;\ sd \leftarrow sc, not\ d\}$. The answer sets of P^{es} are $M_4 = \{a, sc, sd\}$ and $M_5 = \{sb, sc, sd\}$. Note that M_5 is not a minimal extended externally supported model of P, because $s(M_4) = \{sc, sd\} \subset s(M_5) = M_5$. Therefore, $MEES(P) = \{M_4\}$.

The *MES*-semantics is similar to the semi-stable one. Indeed, informally, each minimal externally supported model corresponds to a semi-stable model that has the same true atoms and the same false atoms (once supporting atoms are ignored). The next theorems provide a formal result.

Theorem 3. Let P be a program, and let $M \in MES(P)$. Then, there exists $M' \in SST(P)$, such that (i) $M \setminus s(M) = M' \setminus \kappa(M')$, (ii) if $sa \in M$, then $Ka \in M'$; and (iii) if $Ka \in M'$ and $sa \notin M$, then $a \in M'$.

Proof. Let M be a minimal externally supported model of P. Starting from M, we build an M' that is a semi-stable model of P. First of all, we consider an interpretation I satisfying the first two conditions, (i) and (ii), of the theorem. We show that I is a model of the epistemic transformation P^κ. Note that each positive rule of P^s is also a positive rule of P^κ. Hence, I satisfies it, because $I \setminus \kappa(I) = M \setminus s(M)$ and $M \setminus s(M)$, by assumption, satisfies it. Now, if I satisfies all rules of P^κ, then we are done. Otherwise, suppose that I does not satisfy a rule of the form (2). (Note that by construction of I, no $\lambda_{r,i}$ belongs to I, hence all bodies of the rules of the form (3)-(5) are false.) Therefore, there exists $r \in P$ such that $b_1, \ldots, b_m \in I$ and $\lambda_{r,1}, \ldots, \lambda_{r,l}, Kc_1, \ldots, Kc_n \notin I$. Hence, as $M \models r^s$, either (a) there exists $c_j \in M$, for some $j \in \{1, \ldots, n\}$, or (b) there exists $a_i \in M$, for some $i \in \{1, \ldots, l\}$. In the case (a), we add Kc_j in I, so $I \cup \{Kc_j\}$ satisfies all rules in the epistemic transformation of r. In the case (b), we add $\lambda_{r,i}$ in I, so $I \cup \{\lambda_{r,i}\}$ satisfies the first two rules and the last rule in the epistemic transformation of r.

Suppose, by contradiction, that $I \cup \{\lambda_{r,i}\}$ does not satisfy the rule of the form (4), that is, there exists $j \in \{1, \ldots, n\}$ such that $c_j \in I$. Therefore, by construction, $c_j \in M$. Hence, we are with case (a) really, covered earlier. Now, the model M' so constructed (up to now called I) satisfies the third condition of the theorem and is also minimal. Indeed, during the construction of M', we have added atoms of the form Kc (only if c was in M) or λ, only when necessary. Hence, if a is such a atom, then $M' \setminus \{a\}$ is not a model. Concerning the other atoms, they are also in M, so if we delete one of these from M', we could delete the same atom from M, contradicting the minimality of M. Moreover, if an atom Kc has been added in M', and sc was not in M, then c is in M', as c was in M. Hence, the third condition is satisfied, and M' is a minimal model of P^κ. Finally, it is easy to see that the minimization of the atoms of the form sc in M induces the corresponding minimization of the gap atoms in M'. Therefore, M' is a maximal canonical interpretation among the minimal models of P^κ, that is a semi-stable model of P. □

Theorem 4. Let P be a program, and let $M \in SST(P)$. Then, there exists $M' \in MES(P)$, such that (i) $M' \setminus s(M') = M \setminus \kappa(M)$, and (ii) $Ka \in \mathscr{G}(M)$ if, and only if, $sa \in M'$.

Proof. Let $M \in SST(P)$. Then, we consider the following interpretation over Σ^s: $M' = (M \setminus \kappa(M)) \cup \{sa : Ka \in \mathscr{G}(M)\}$. Clearly, by construction, M' satisfies condition (i). Indeed, $M' \setminus s(M') = M \setminus \kappa(M)$. And, moreover, M' satisfies condition (ii). Indeed, $\{sa : Ka \in \mathscr{G}(M)\}$ is the set of all support atoms in M'. Therefore, it remains to prove that M' is a minimal extended supported model of P. First, we show that M' is a model of P^s. Clearly, M' satisfies all rules of the form $\{sc\}$. Now, assume by contradiction that there is a rule in P^s of the form (8)

that is not satisfied by M'. That is, $b_1,\ldots,b_m \in M'$ and $c_1,\ldots,c_n,sc_1,\ldots,sc_n,a_1,\ldots,a_l \notin M'$. Let S be an answer set of P^κ such that $M = S \cap \Sigma^\kappa$. Then, by construction of M', we have that $b_1,\ldots,b_m \in S$ and $c_1,\ldots,c_n,a_1,\ldots,a_l \notin S$ (by condition (i)); and $Kc_1,\ldots,Kc_n \notin S$ (by condition (ii)). Therefore, consider the corresponding rule of the form (2). As S satisfies it, then some $\lambda_{r,i} \in S$, but $a_i \notin S$. So S does not satisfy the rule of the form (3), in conflict with S being a model of P^κ. Therefore, M' is a model of P^s.

To show that M' is an answer set of P^s, we assume by contradiction that there is a model I' of the reduct of P^s with respect to M', $P^{sM'}$, such that I' is strictly contained in M'. Note that $s(I')$ must be equal to $s(M')$, as it satisfies the choice rules. Now, assume that for some rule of the form (8) $\{a_1,\ldots,a_l\} \cap I' \subset \{a_1,\ldots,a_l\} \cap M'$. Then, we consider the corresponding interpretation over Σ^κ, that is $I = S \setminus (M' \setminus I')$, and then we consider $J = I \cup \{\lambda_{r,i} : a_i \in I\}$, so that J is strictly contained in S, and J by construction satisfies rules of the form (3) and (5). Moreover, J cannot be a model of P^κ, as S is a minimal model of P^κ. Therefore J does not satisfy some rule of the form (2) or of the form (4). First, suppose that a rule of the form (4) is not satisfied by J. Hence, $\lambda_{r,i}, c_j \in J$. Then, $\lambda_{r,i}, c_j \in S$, thus S would not be a model. Now suppose that a rule of the form (2) is not satisfied by J. This means that $b_1,\ldots,b_m \in J$, but $\lambda_{r,1},\ldots,\lambda_{r,l},Kc_1,\ldots,Kc_n \notin J$. Hence, also $a_1,\ldots,a_l \notin J$, and further $b_1,\ldots,b_m \in I'$, $sc_1,\ldots,sc_n \notin I'$, and $a_1,\ldots,a_l \notin I'$. Therefore, there exists some $c_j \in I'$ to satisfy the corresponding rule of the form (8). Therefore, $c_j \in M'$, thus this rule is not in $P^{sM'}$. Hence, $\{a_1,\ldots,a_l\} \cap I' = \{a_1,\ldots,a_l\} \cap M'$, for each rule of the form (8). Therefore, $I' \not\subset M'$, that is, M' is an answer set of P^s.

Finally, by the gap minimality of M, it follows that the number of support atoms is also minimal in M', that is, M' is a minimal externally supported model of P^s. □

Note that a minimal externally supported model can correspond to more than one semi-stable model.

Example 2. Consider the program $P = \{a;\ b;\ c \leftarrow not\ a, not\ b;\ d \leftarrow not\ a, not\ b;\ c \leftarrow not\ c\}$. It is incoherent, with a single minimal externally supported model $\{a,b,sc\}$ of P, while $SST(P) = \{\{a,b,Kc,Ka\},\{a,b,Kc,Kb\}\}$. Note that both $\{a,b,Kc,Ka\}$ and $\{a,b,Kc,Kb\}$ can be obtained starting from $\{a,b,sc\}$, as expected by Theorem 4.

The *MEES*-semantics is very close to the semi-equilibrium semantics. Informally, each minimal extended externally supported model corresponds to a semi-equilibrium model that has the same true atoms and the same false atoms, and vice versa. Indeed, with a slight modification of the proof strategy used to prove Theorems 3 and 4, it can be shown that there is a one-to-one correspondence between minimal extended externally supported models and semi-equilibrium models, as stated in the following theorem.

Theorem 5. Let P be a program. Then, $M \in MEES(P)$ if, and only if, $M' \in SEQ(P)$, where $M \setminus s(M) = M' \setminus \kappa(M')$, and $sa \in M$ if, and only if, $Ka \in \mathscr{G}(M')$.

Example 3. Consider again the program P of Example 2. It has a single minimal extended externally supported model equal to the minimal externally supported model, i.e., $MEES(P) = \{\{a,b,sc\}\}$. Moreover, it also has a single semi-equilibrium model: $SEQ(P) = \{\{a,b,Ka,Kb,Kc\}\}$.

Theorems 3-4 and Theorem 5, basically, say that computing minimal externally supported models and minimal extended externally supported models corresponds to computing semi-stable and semi-equilibrium models, respectively.

4 Computation of Paracoherent Answer Sets

We now discuss how to effectively compute one paracoherent answer set for a program, by using the (extended) externally supported semantics. We first discuss some algorithms of (Amendola et al. 2017), because it is possible to re-use them. These algorithms take as input a program $\Pi = P^\chi \cup P^{gap}$, where P^χ is a generic epistemic transformation of the ASP program P and P^{gap} is the following set of rules capturing the notion of the gap:

$$gap(Ka) \leftarrow Ka,\ not\ a; \qquad \forall a \in \Sigma \qquad (11)$$

Let $gap(I) = \{gap(Ka) \mid gap(Ka) \in I\}$, for a set I of atoms. An answer set M of Π is a paracoherent answer set of P if, and only if, there exists no answer set M_1 of Π such that $gap(M_1) \subset gap(M)$. Based on this, a set of algorithms for computing a subset minimal (with respect to the gap atoms) answer set was proposed in (Amendola et al. 2017). We briefly recall the three algorithms obtaining the best performance, namely MINIM, SPLIT, and WEAK.

The idea of MINIM is to compute an answer set M of Π and then to search for another answer set M^w such that $gap(M^w) \subset gap(M)$. This property is enforced by a set of rules added to Π. If Π admits an answer set, say M^w, then M is replaced by M^w and the algorithm iterates minimizing M. Otherwise, if Π admits no answer set, M is a paracoherent answer set and the algorithm terminates returning M.

Concerning SPLIT, the algorithm first computes an answer set M of Π and creates a set C of gap atoms that are included in M. The program Π is then modified by adding the constraint $\leftarrow p$ for all gap atoms that are not included in M. Subsequently, one of the atoms in C is randomly selected, say a, and an answer set of $\Pi \cup \{\leftarrow a\}$ is searched. If such an answer set does not exist then a must be included in the paracoherent answer set. Thus, Π is modified by adding the constraint $\leftarrow not\ a$ and a is removed from the set C. Otherwise, if $\Pi \cup \{\leftarrow a\}$ admits an answer set, say M^w, then M is replaced by M^w and the set C is replaced by the gap atoms that are true in M^w. The algorithm iterates until C is empty, returning M corresponding to the paracoherent answer set.

Algorithm WEAK is based on the observation that modern ASP solvers are able to compute cardinality minimal answer set w.r.t. a set of atoms using the so-called *weak constraints*. Therefore, a cardinality minimal answer set with respect to the gap atoms is also subset minimal with respect to the gap atoms, and so, it is a paracoherent answer set of P.

As observed in the previous section, given a program P, a paracoherent answer set of P is an answer set of P^s (resp. P^{es}) that is subset minimal with respect to the atoms of the form sc (for $c \in \Sigma$). Thus, algorithms proposed in (Amendola et al. 2017) can be used also in our setting. In particular,

the algorithms take as input the transformation P^s (resp. P^{es}) and produce as output an answer set that is subset minimal with respect to the atoms of the form sc. It is worth mentioning that P^s (resp. P^{es}) can be more compact than P^κ (resp. P^{HT}). Let $rules(P)$ and $atoms(P)$ be the set of rules of the form (1) and the set of atoms occurring in P, respectively.

Proposition 1. *Given a program P, $|rules(P^\kappa)| \geq |rules(P^s)|$ and $|atoms(P^\kappa)| \geq |atoms(P^s)|$.*

Proof. $|rules(P^\kappa)| = |rules(P)| + \sum_{r\in P}|H(r)| + \sum_{r\in P}|B^-(r)| + \sum_{r\in P}|H(r)|^2$ while $|rules(P^s)| = |rules(P)| + |\bigcup_{r\in P}B^-(r)|$. Thus, $|rules(P^\kappa)| \geq |rules(P^s)|$ because $\sum_{r\in P}|B^-(r)| \geq |\bigcup_{r\in P}B^-(r)|$. $|atoms(P^\kappa)| = |atoms(P)| + \sum_{r\in P}|H(r)| + |\bigcup_{r\in P}B^-(r)|$ while $|atoms(P^s)| = |atoms(P)| + |\bigcup_{r\in P}B^-(r)|$. Thus, $|atoms(P^\kappa)| \geq |atoms(P^s)|$. □

Proposition 2. *Given a program P, $|rules(P^{HT})| \geq |rules(P^{es})|$ and $|atoms(P^{HT})| \geq |atoms(P^{es})|$.*

Proof. $|rules(P^{HT})| = |rules(P^\kappa)| + |atoms(P)| + |rules(P)|$ while $|rules(P^{es})| = |rules(P^s)| + |rules(P)|$. Thus, $|rules(P^{HT})| \geq |rules(P^{es})|$ because $|rules(P^\kappa)| \geq |rules(P^s)|$. $|atoms(P^{HT})| = 2 \times |atoms(P)| + \sum_{r\in P}|H(r)|$ while $|atoms(P^{es})| = 2 \times |atoms(P)|$, whence $|atoms(P^{HT})| \geq |atoms(P^{es})|$. □

5 Implementation and Experiments

In this section, we present an experimental analysis conducted to study the impact of the new strategy for computing a paracoherent answer set based on minimal (extended) externally supported models.

Implementation. Algorithms described in the previous section were implemented using the state of the art ASP solver WASP (Alviano et al. 2015). In particular, we reused the variant of WASP presented in (Amendola et al. 2017). The implementation is parametric w.r.t. the transformations, and makes use of precisely the same rewriter tools to produce either the epistemic transformation or the (extended) externally supported program.

Benchmark Settings. Experiments were run on a system with 2.30GHz Intel Xeon E5-4610 v2 CPUs. Execution time and memory were limited to 1200 seconds and 8 GB, respectively. We used exactly the same benchmark from (Amendola et al. 2017). That benchmark addresses debugging, 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. The instances are from the latest ASP competition (Gebser, Maratea, and Ricca 2015), which implies that they are challenging for ASP solvers. The selected benchmarks are Knight Tour with Holes, Minimal Diagnosis, Qualitative Spatial Reasoning, Stable Marriage, and Visit All. Basically, the selection includes all the incoherent instances in the competition suite that feature neither *aggregates*, nor *choice rules*, nor *weak constraints*, since such features are not currently supported by the paracoherent semantics (Amendola et al. 2016b).

Results. The experiments show a clear advantage of using the novel translations, in the vast majority of considered instances the improvements are significant, as seen from the cactus plots in Figures 1(a) and 1(b) and the scatter plots in Figures 1(c) and 1(d). The major factor for the improvements appears to be the smaller size of the resulting programs, which lead to decreased memory usage, as seen in the scatter plots in Figures 1(e) and 1(f).

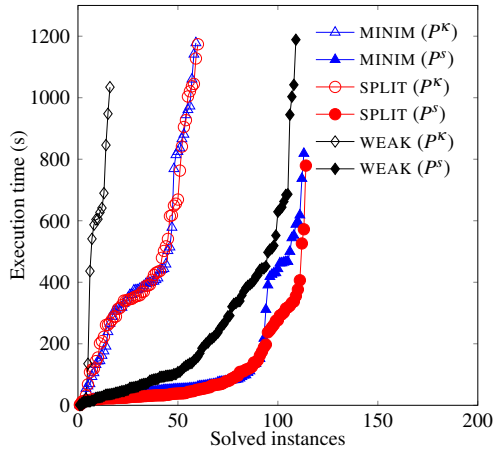
In more detail, Figure 1(a) shows that WASP executed on the program P^s outperformed WASP executed on P^κ, for all considered algorithms (ALG(P) means WASP running the algorithm ALG on the program P). Also for P^{es} and P^{HT}, we observed an advantage when WASP is executed on the former, seen in Figure 1(b). Indeed, MINIM (P^{es}) and SPLIT (P^{es}) solve 5 and 2 more instances than their counterparts executed on P^{HT}. A huge improvement of the performance is instead observed when the algorithm WEAK is considered. Indeed, WEAK (P^{es}) solves 60 more instances than WEAK (P^{HT}). There is a somewhat peculiar characteristic for the algorithm WEAK. Indeed, WEAK (P^{es}) by far outperforms both MINIMP^{es} and SPLITP^{es}, while the opposite happens when the program P^s is considered. Upon closer look, this advantage is mostly due to the benchmark Minimal Diagnosis, where the default core-based strategy of WASP (Alviano and Dodaro 2016) (used by algorithm WEAK) terminates after one call to the solver.

Figures 1(c) and 1(d) show instance-by-instance comparisons for systems using the old and new translations. If for an instance the two systems take x and y execution time, then a point (x,y) is plotted. Therefore, points below the diagonals represent instances where the system reported on the x-axis was slower than the system reported on the y-axis. The graphs confirm the better performance of WASP executed on P^s (resp. P^{es}) than its counterparts executed on P^κ (resp. P^{HT}), independently of the used algorithm. Indeed, only few instances are on the left of the diagonals, meaning that are only few instances where WASP executed on P^s (resp. P^{es}) is slower than WASP executed on P^κ (resp. P^{HT}).

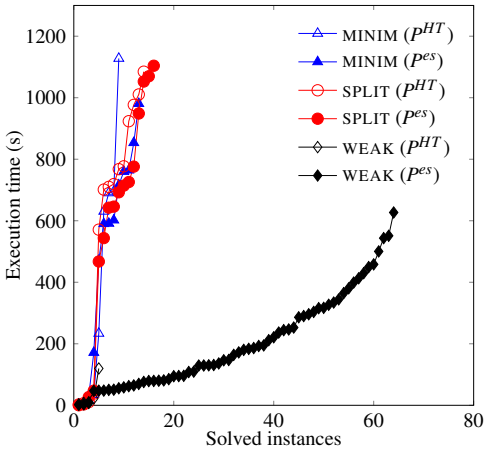
The main reason for this advantage appears to be due to the different sizes of programs processed by WASP. Looking at Table 1, we observe that the average numbers of atoms and of rules for each benchmark are much lower for P^s (resp. P^{es}) than for P^κ (resp. P^{HT}) (cf. also Propositions 1 and 2). The smaller programs impact the memory usage of WASP as the instance-wise comparison in Figures 1(e) and 1(f) clearly shows: WASP uses consistently less memory with P^s (resp. P^{es}) than with P^κ (resp. P^{HT}).

6 Related Work

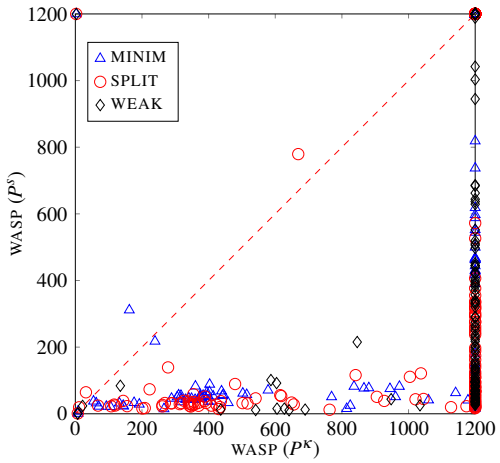
In this paper we focus on the computation of the semi-stable (Sakama and Inoue 1995) and the semi-equilibrium semantics (Amendola et al. 2016b). These semantics are considered the two major paracoherent semantics for ASP programs, that emerged over several alternative proposals (Przymusinski 1991; van Gelder, Ross, and Schlipf 1991;



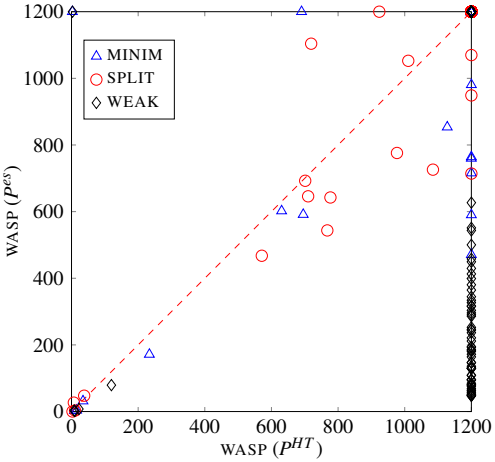
(a) Comparison of all algorithms on P^{κ} and P^{s}.



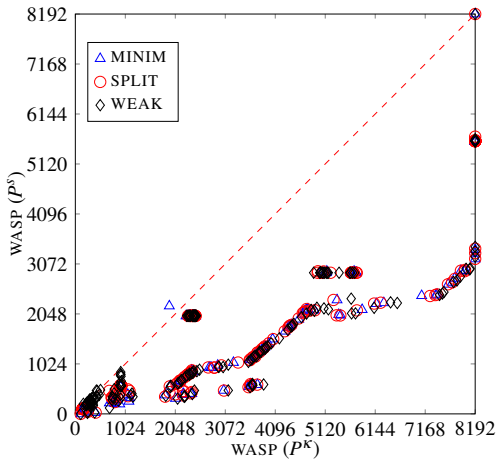
(b) Comparison of all algorithms on P^{HT} and P^{es}.



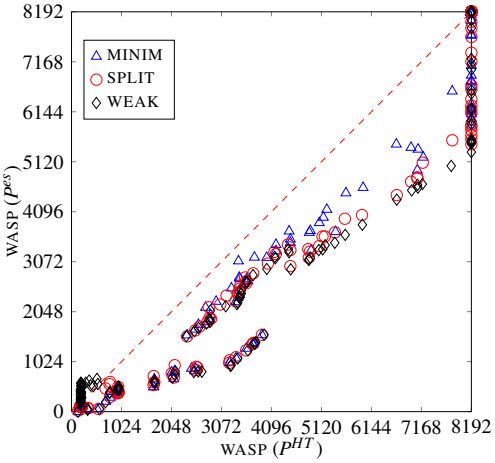
(c) Instance-wise comparison of all algorithms implemented in WASP and executed on P^{κ} and P^{s}.



(d) Instance-wise comparison of all algorithms implemented in WASP and executed on P^{HT} and P^{es}.



(e) Instance-wise comparison of memory usage of algorithms implemented in WASP and executed on P^{κ} and P^{s}.



(f) Instance-wise comparison of memory usage of algorithms implemented in WASP and executed on P^{HT} and P^{es}.

Figure 1: Comparison of the performance of WASP executed on the programs P^{κ}, P^{s}, P^{HT}, and P^{es} produced by the different rewriting techniques.

Benchmarks	#	P^{κ}		P^{s}		P^{HT}		P^{es}	
		Atoms	**Rules**	**Atoms**	**Rules**	**Atoms**	**Rules**	**Atoms**	**Rules**
KnightTour	26	358 035	655 709	141 054	474 852	358 035	1 238 936	167 067	906 915
MinDiagn	64	1 214 042	1 579 409	450 265	654 412	1 214 042	2 549 729	466 578	1 216 487
QualSpatReas	76	68 793	789 642	27 620	735 317	68 793	1 542 607	28 035	1 469 246
StableMarr	1	-	-	-	-	-	-	-	-
VisitAll	5	13 926	72 776	4 650	64 234	13 926	140 881	7 567	128 342

Table 1: Impact of transformations P^{κ}, P^{s}, P^{HT}, and P^{es}.

Eiter, Leone, and Saccà 1997; Seipel 1997; Balduccini and Gelfond 2003; Pereira and Pinto 2005; Alcântara, Damásio, and Pereira 2005; Galindo, Ramírez, and Carballido 2008), because they satisfy all the following five highly desirable –from the knowledge representation point of view– theoretical properties (Amendola et al. 2016b): *answer set coverage*, *congruence*, *classical coherence*, *minimal undefinedness* and *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 (such a property has been recently enforced in Fuzzy ASP (Alviano, Amendola, and Peñaloza 2017)), and every true atom should be derived from the program, respectively. The partial evidential stable models of (Seipel 1997) are known to be equivalent to semi-equilibrium ones (Amendola et al. 2016b), thus our characterization can be used for implementing this semantics. Moreover, in (Amendola, Eiter, and Leone 2014), to refine the semi-equilibrium semantics for theoretical reasons a splitting technique has been proposed. This technique could have some potential to be used to improve the performance, as it could reduce the size of a subcomponent passed to a solver. However, it could not find any semi-equilibrium model. Finally, we mention that the notion of *support* used in this work has a long history, from the application to general logic programs (Fages 1991) up to the recent use in ontological reasoning (Amendola, Leone, and Manna 2017).

The efficient computation of paracoherent semantics for ASP programs has been tackled only recently in (Amendola et al. 2017). There, a number of algorithms relying on the epistemic transformation of programs have been proposed. We build upon that work, and present a transformation that can replace the epistemic transformation in the evaluation pipeline. As discussed in Section 4, the new transformation introduces fewer atoms and fewer rules. Moreover, our approach significantly improve the performance of solvers based on the algorithms presented in (Amendola et al. 2017) as shown in Section 5. The algorithms used for computing paracoherent answer sets are strictly related to the computation of minimal models of propositional theories. The first approaches to that problem (Niemelä 1996; Hasegawa, Fujita, and Koshimura 2000; Bry and Yahya 2000) were not able to take profit of modern learning-based algorithms (Koshimura et al. 2009). Later the first algorithm able to overcome that technological limit for the computation of minimal models of SAT formulae was introduced in (Koshimura et al. 2009) that is similar in principle to the MINIM algorithm; whereas the SPLIT algorithm is similar to the algorithms employed for computing cautious consequences of ASP programs (Alviano, Dodaro, and Ricca 2014) and backbones of SAT formulas (Janota, Lynce, and Marques-Silva 2015). General approaches such as the algorithms for computing a Minimal Set over a Monotone Predicate (MSMP) (Janota and Marques-Silva 2016) could be adapted for computing paracoherent answer sets; whereas the algorithms for positive CNF theories of (Angiulli et al. 2014) cannot be applied, since the minimization of the extension of a specific predicate is not supported.

7 Discussion and Conclusion

The computation of a paracoherent answer set is a difficult problem that has been addressed only recently (Amendola et al. 2017). Existing implementations of this task work according to the definition of paracoherent answer sets, calling an answer set solver repeatedly for finding a maximal canonical answer set of the epistemic transformation.

In this paper we presented an alternative characterization of the two major paracoherent semantics in terms of (extended) externally supported models. The new definition is based on a transformation of the program that can replace the epistemic transformation in existing strategies for computing semi-stable and semi-equilibrium models. We developed concrete implementations that use the new transformation associated with algorithms of (Amendola et al. 2017). An experimental analysis carried out on benchmarks from ASP competitions shows that the new transformation brings huge performance improvements that are independent of the underlying algorithms. The improvements can be explained by observing that the new transformation is more parsimonious than the epistemic transformation in the sense that it introduces fewer auxiliary propositional atoms and rules. The ideas presented in this paper represent a significant step towards in the state of the art of methods for computing paracoherent answer sets, which opens new possibilities for implementing concrete applications of paracoherent semantics.

Acknowledgements

This work has been partially supported by the EU Horizon 2020 Marie Skłodowska-Curie grant agreement No. 690974 for the project "MIREL", and by the Italian ministry for economic development (MISE) under project "PIUCultura" (n. F/020016/01-02/X27) and under project "S2BDW" (n. F/050389/01-03/X32).

References

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

Alviano, M.; Amendola, G.; and Peñaloza, R. 2017. Minimal undefinedness for fuzzy answer sets. In *AAAI 2017*, 3694–3700.

Alviano, M., and Dodaro, C. 2016. Anytime answer set optimization via unsatisfiable core shrinking. *TPLP* 16(5-6):533–551.

Alviano, M.; Dodaro, C.; Leone, N.; and Ricca, F. 2015. Advances in WASP. In *LPNMR 2015*, 40–54.

Alviano, M.; Dodaro, C.; and Ricca, F. 2014. Anytime computation of cautious consequences in answer set programming. *TPLP* 14(4-5):755–770.

Amendola, G.; Dodaro, C.; Leone, N.; and Ricca, F. 2016a. On the application of answer set programming to the conference paper assignment problem. In *AI*IA 2016*, 164–178.

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

Amendola, G.; Greco, G.; Leone, N.; and Veltri, P. 2016c. Modeling and reasoning about NTU games via answer set programming. In *IJCAI 2016*, 38–45.

Amendola, G.; Dodaro, C.; Faber, W.; Leone, N.; and Ricca, F. 2017. On the computation of paracoherent answer sets. In *AAAI 2017*, 879–885.

Amendola, G.; Eiter, T.; and Leone, N. 2014. Modular paracoherent answer sets. In *JELIA 2014*, 457–471.

Amendola, G.; Leone, N.; and Manna, M. 2017. Finite model reasoning over existential rules. *TPLP* 17(5-6):726–743.

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

Balduccini, M., and Gelfond, M. 2003. Logic programs with consistency-restoring rules. In *ISLFCR, AAAI 2003 Spring Symposium Series*, 9–18.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Com. ACM* 54(12):92–103.

Bry, F., and Yahya, A. H. 2000. Positive unit hyperresolution tableaux and their application to minimal model generation. *J. Autom. Reasoning* 25(1):35–82.

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). *TPLP* 16(5-6):653–669.

Eiter, T.; Leone, N.; and Saccà, D. 1997. On the partial semantics for disjunctive deductive databases. *Ann. Math. Artif. Intell.* 19(1-2):59–96.

Fages, F. 1991. A new fixpoint semantics for general logic programs compared with the well-founded and the stable model semantics. *New Generation Comput.* 9(3/4):425–444.

Gaggl, S. A.; Manthey, N.; Ronca, A.; Wallner, J. P.; and Woltran, S. 2015. Improved answer-set programming encodings for abstract argumentation. *TPLP* 15(4-5):434–448.

Galindo, M. J. O.; Ramírez, J. R. A.; and Carballido, J. L. 2008. Logical weak completions of paraconsistent logics. *J. Log. Comput.* 18(6):913–940.

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

Gebser, M.; Maratea, M.; and Ricca, F. 2015. The design of the sixth answer set programming competition - report -. In *LPNMR 2015*, 531–544.

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

Grasso, G.; Leone, N.; Manna, M.; and Ricca, F. 2011. ASP at work: Spin-off and applications of the DLV system. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, LNCS 6565, 432–451.

Hasegawa, R.; Fujita, H.; and Koshimura, M. 2000. Efficient minimal model generation using branching lemmas. In *CADE-17, 2000*, 184–199.

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

Janota, M.; Lynce, I.; and Marques-Silva, J. 2015. Algorithms for computing backbones of propositional formulae. *AI Commun.* 28(2):161–177.

Koshimura, M.; Nabeshima, H.; Fujita, H.; and Hasegawa, R. 2009. Minimal model generation with respect to an atom set. In *FTP 2009*, CEUR 556.

Manna, M.; Ricca, F.; and Terracina, G. 2015. Taming primary key violations to query large inconsistent data via ASP. *TPLP* 15(4-5):696–710.

Niemelä, I. 1996. A tableau calculus for minimal model reasoning. In *TABLEAUX 1996*, 278–294.

Pereira, L. M., and Pinto, A. M. 2005. Revised stable models - a semantics for logic programs. In *EPIA*, 29–42.

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

Sakama, C., and Inoue, K. 1995. Paraconsistent stable semantics for extended disjunctive programs. *J. Log. Comput.* 5(3):265–285.

Seipel, D. 1997. Partial evidential stable models for disjunctive deductive databases. In Dix, J.; Pereira, L. M.; and Przymusinski, T. C., eds., *LPKR*, volume 1471 of *LNCS*, 66–84. Springer.

Simons, P.; Niemelä, I.; and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artif. Intell.* 138(1-2):181–234.

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.