Solving the Stable Roommates Problem using Incoherent Answer Set Programs

Giovanni Amendola

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

Abstract. Answer Set Programming (ASP) has become an established logic-based programming paradigm with successful applications. In this paper, through the study of a direct and natural modeling of the Stable Marriage Problem (SMP) via ASP, we apply the same approach to the Stable Roommates Problem (SRP). However, unlike the SMP, the modeling proposed may lead to lack of answer sets due to cyclic default negation occurring in the ASP program. Hence, the proposed modeling of the SRP can lead to a first benchmark in the ASP competions with consistent, but incoherent ASP programs.

Keywords: Answer Set Programming · Paracoherent Resoning · Semi-Equilibrium Models · Stable Marriage Problem · Stable Roommates Problem.

1 Introduction

Answer Set Programming (ASP) is a premier formalism for knowledge representation and non-monotonic reasoning. It is a declarative programming paradigm oriented towards difficult search problems. Indeed, ASP combines a comparatively high knowledge-modeling power [16, 2, 15] with a robust solving technology [17, 23–26, 33, 8, 12–14, 37, 36]. The idea of ASP is to represent a given computational problem by a logic program whose answer sets correspond to solutions, and then use a solver to find them. For these reasons ASP has become an established logic-based programming paradigm with successful applications to complex problems in several areas, such as Artificial Intelligence [29, 28, 7, 4], Bioinformatics [21], Databases [35], Game Theory [11], Information Extraction [1].

Recently, ASP programs have been used to encode a number of variations and generalizations of the Stable Marriage Problem (SMP) [19]. The SMP requires to find a way to arrange the marriage for the men and women with respect to mutual preferences [31]. Given a set of men M and a set of women W of the same size, and a set of preferences, a solution to SMP is a bijective function S from M to W such that there is no pair $(m,w) \in M \times W$, where m prefers w to $S(m)$, and w prefers m to $S^{-1}(w)$. It is well-known that it is always possible to solve the SMP and make all marriages stable [22]. The SMP has been addressed from an abstract argumentation perspective by Dung in its pioneering work [20].

In this paper, first we show that the modeling approach proposed by Dung can be ported to ASP, and it represents a direct and natural encoding of the SMP. The main feature of this modeling is the absence of constraints in the logic program. Then, we consider a variation of the SMP, known as the Stable Roommates Problem (SRP). Given a set of $2n$ persons, each one ranks the others in strict order of preference. A solution to the SRP is a set of n disjoint pairs of persons so that there are no two persons p_1 and p_2, each of whom prefers the other to their partner. As the SRP has a structure similar to the SMP, the modeling of the SMP via ASP remains a direct and natural encoding for the SRP. However, unlike the SMP, a solution to the SRP may fail to exist for certain sets of persons and their preferences, and so no answer set could exist.

Since no constraint appears in the encoding, the lack of answer sets is due to the cyclic default negation. It is noteworthy to mention that in all the benchmarks appearing in the ASP competitions [26], the lack of answer sets is always due to the violation of some constraint. Hence, the proposed modeling of the SRP can lead to have a first benchmark with consistent, but incoherent ASP programs.

2 Preliminaries

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

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

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

Any subset I of Σ is an *interpretation*. An interpretation I is a *model* of a program P (denoted $I \models P$) if and only if for each rule $r \in P$, $I \cap H(r) \neq \emptyset$ if $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$ (denoted $I \models r$). A model M of P is *minimal*, if and only if there is no model M' of P such that $M' \subset M$. We denote by $MM(P)$ the set of all minimal models of P. Given an interpretation I, let P^I be the well-known *Gelfond-Lifschitz reduct* [27] of P with respect to I, i.e., the set of rules $a_1 \vee ... \vee a_l \leftarrow b_1, ..., b_m$, obtained from rules $r \in P$ of form (1), such that $B^-(r) \cap I = \emptyset$. An interpretation I is an *answer set* of P if $I \in MM(P^I)$. We denote by $AS(P)$ the set of all answer sets (called also *stable models*) of P. Finally, we say that a program P is *consistent*, if it admits some model, otherwise it is *inconsistent*; whereas we say that it is *coherent*, if it admits some answer set (i.e., $AS(P) \neq \emptyset$), otherwise, it is *incoherent* [3, 10, 9].

Example 1. Consider the following logic program

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

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

3 Stable Marriage Problem: from Argumentation to ASP

The Stable Marriage Problem (SMP) requires to find a way to arrange the marriage for the men and women with respect to mutual preferences [31]. Given a set M of n men, a set W of n women, and a set of preferences of the form $m \in M$ *prefers* $w_1 \in W$ *to* $w_2 \in W$ or $w \in W$ *prefers* $m_1 \in M$ *to* $m_2 \in M$, a solution to SMP is a bijective function S from M to W such that there is no pair $(m,w) \in M \times W$, where m prefers w to $S(m)$, and w prefers m to $S^{-1}(w)$. Note that, it is implicitely assumed that $M \cap W = \emptyset$. It is well-known that it is always possible to solve the SMP and make all marriages stable [22].

The SMP has been addressed from an abstract argumentation perspective by Dung in its pioneering work [20]. An argumentation framework AF is defined as (Ar, att), where Ar is a set of arguments and $att \subseteq Ar \times Ar$ is a set of attacks. For instance, if $AF = (\{a,b,c\},\{(a,b),(b,c)\})$, then argument a attacks argument b, and argument b attacks argument c. Dung introduced the so-called *stable semantics* for argumentation framework. A set of arguments A is a stable extension, if (1) each argument in A does not attack an argument in A; and (2) each argument outside A is attacked by some argument in A. For instance, if we consider the previous argumentation framework AF, we have that $A = \{a,c\}$ is a stable extension. Indeed, (1) (a,c) and (c,a) do not belong to att; and (2) the argument b (not in A) is attacked by $a \in A$.

In [20], Dung proposed a modeling of the SMP via abstract argumentation framework. Starting from M, W, and a set of preferences, defined as above, he constructs an argumentation framework $AF = (Ar, att)$, where $Ar = M \times W$, and a pair $(m_1,w_1) \in M \times W$ attacks $(m_2,w_2) \in M \times W$ if, and only if, (i) $m_1 = m_2$ and m_1 prefers w_1 to w_2; or (ii) $w_1 = w_2$ and w_1 prefers m_1 to m_2. Then, he was able to prove that

Theorem 1 (Theorem 39 in [20]). *A set $S \subseteq M \times W$ constitutes a solution to the SMP if, and only if, S is a stable extension of the corresponding argumentation framework.*

Example 2. Consider the following instance of the SMP: $M = \{m_1,m_2,m_3,m_4\}$ and $W = \{w_1,w_2,w_3,w_4\}$ with the basic prefence relations:

m_1 prefers w_2 to w_4;	m_1 prefers w_4 to w_1;	m_1 prefers w_1 to w_3;
m_2 prefers w_3 to w_1;	m_2 prefers w_1 to w_4;	m_2 prefers w_4 to w_2;
m_3 prefers w_2 to w_3;	m_3 prefers w_3 to w_1;	m_3 prefers w_1 to w_4;
m_4 prefers w_4 to w_1;	m_4 prefers w_1 to w_3;	m_4 prefers w_3 to w_2;
w_1 prefers m_2 to m_1;	w_1 prefers m_1 to m_4;	w_1 prefers m_4 to m_3;
w_2 prefers m_4 to m_3;	w_2 prefers m_3 to m_1;	w_2 prefers m_1 to m_2;

w_3 prefers m_1 to m_4; w_3 prefers m_4 to m_3; w_3 prefers m_3 to m_2;
w_4 prefers m_2 to m_1; w_4 prefers m_1 to m_4; w_4 prefers m_4 to m_3.

We reported basic preference relations only. However, they imply others. For instance, from m_1 prefers w_2 to w_4 and m_1 prefers w_4 to w_1, one can deduce that m_1 prefers w_2 to w_1. Hence, the corresponding argumentation framework is formed by $Ar = M \times W$ and

$$att = \left\{ \begin{array}{l} ((m_1,w_2),(m_1,w_4)),\ ((m_1,w_4),(m_1,w_1)),\ ((m_1,w_1),(m_1,w_3)) \\ ((m_2,w_3),(m_2,w_1)),\ ((m_2,w_1),(m_2,w_4)),\ ((m_2,w_4),(m_2,w_2)) \\ ((m_3,w_2),(m_3,w_3)),\ ((m_3,w_3),(m_3,w_1)),\ ((m_3,w_1),(m_3,w_4)) \\ ((m_4,w_4),(m_4,w_1)),\ ((m_4,w_1),(m_4,w_3)),\ ((m_4,w_3),(m_4,w_2)) \\ ((m_2,w_1),(m_1,w_1)),\ ((m_1,w_1),(m_4,w_1)),\ ((m_4,w_1),(m_3,w_1)) \\ ((m_4,w_2),(m_3,w_2)),\ ((m_3,w_2),(m_1,w_2)),\ ((m_1,w_2),(m_2,w_2)) \\ ((m_1,w_3),(m_4,w_3)),\ ((m_4,w_3),(m_3,w_3)),\ ((m_3,w_3),(m_2,w_3)) \\ ((m_2,w_4),(m_1,w_4)),\ ((m_1,w_4),(m_4,w_4)),\ ((m_4,w_4),(m_3,w_4)) \\ ((m_1,w_2),(m_1,w_1)),\ ((m_1,w_2),(m_1,w_3)),\ ((m_1,w_4),(m_1,w_3)) \\ ((m_2,w_3),(m_2,w_4)),\ ((m_2,w_3),(m_2,w_2)),\ ((m_2,w_1),(m_2,w_2)) \\ ((m_3,w_2),(m_3,w_1)),\ ((m_3,w_2),(m_3,w_4)),\ ((m_3,w_3),(m_3,w_4)) \\ ((m_4,w_4),(m_4,w_3)),\ ((m_4,w_4),(m_4,w_2)),\ ((m_4,w_1),(m_4,w_2)) \\ ((m_2,w_1),(m_4,w_1)),\ ((m_2,w_1),(m_3,w_1)),\ ((m_1,w_1),(m_3,w_1)) \\ ((m_4,w_2),(m_1,w_2)),\ ((m_4,w_2),(m_2,w_2)),\ ((m_3,w_2),(m_2,w_2)) \\ ((m_1,w_3),(m_3,w_3)),\ ((m_1,w_3),(m_2,w_3)),\ ((m_4,w_3),(m_2,w_3)) \\ ((m_2,w_4),(m_4,w_4)),\ ((m_2,w_4),(m_3,w_4)),\ ((m_1,w_4),(m_3,w_4)) \end{array} \right\}.$$

It can be checked that there are exactly two solutions to the SMP instance: $\{(m_1,w_4)$, (m_2,w_3), (m_3,w_2), $(m_4,w_1)\}$, and $\{(m_1,w_4)$, (m_2,w_1), (m_3,w_2), $(m_4,w_3)\}$, which correspond to the stable extensions of the argumentation framework (Ar, att), according to the Theorem 1.

Recently, relations between abstract argumentation semantics and logic programming semantics has been studied systematically in [18]. These can be highlighted by using a well-known tool for translating argumentation frameworks to logic programs [39]. In particular, given an argumentation framework AF one can build an ASP program, P_{AF} as follows. For each argument a in AF, if c_1, c_2, ..., c_m is the set of its defeaters (i.e., the set of arguments that attack a), we construct the rule

$$a \leftarrow not\, c_1,\ not\, c_2,\ \ldots,\ not\, c_m.$$

Intuitively, each of these rules means that an argument is accepted (inferred as true) if, and only if, all of its defeaters are rejected (inferred as false). More formally, given an argumentation $AF = (Ar, att)$. For each argument $a \in Ar$, we build a rule r_a such that $H(r_a) = \{a\}$, $B^+(r_a) = \emptyset$, and $B^-(r_a) = \{c \in Ar \mid (c,a) \in att\}$. Then, we define P_{AF} as the set of all rules of the form r_a, i.e., $P_{AF} = \{r_a \mid a \in Ar\}$. It is well-known that the answer sets of P_{AF} correspond to the stable extensions of AF [20].

Therefore, the modeling offered by Dung of the SMP through abstract argumentation frameworks, leads to a natural and direct modeling of the SMP through ASP. Starting from M, W, and a set of preferences, defined as above, we constructs an ASP program P as follows. The set of atoms of P is $At(P) = M \times W$. For each pair

$(m,w) \in M \times W$, we build a rule $r_{(m,w)}$ such that $H(r_{(m,w)}) = \{(m,w)\}$, $B^+(r_{(m,w)}) = \emptyset$, and $(m,w') \in B^-(r_{(m,w)})$ if m prefers w' to w; and $(m',w) \in B^-(r_{(m,w)})$ if w prefers m' to m. Hence, P is defined as the set $\{r_{(m,w)} \mid (m,w) \in M \times W\}$. Therefore, it can be shown that

Theorem 2. *A set $S \subseteq M \times W$ constitutes a solution to the SMP if, and only if, S is an answer set of the corresponding ASP program.*

Example 3. Consider again the SMP instance of the Example 2. Hence, the corresponding logic program P is given by the following set of rules:

$$
\begin{aligned}
(m_1,w_1) &\leftarrow \mathit{not}(m_1,w_4),\ \mathit{not}(m_1,w_2),\ \mathit{not}(m_2,w_1)\\
(m_1,w_2) &\leftarrow \mathit{not}(m_3,w_2),\ \mathit{not}(m_4,w_2),\\
(m_1,w_3) &\leftarrow \mathit{not}(m_1,w_1),\ \mathit{not}(m_1,w_4),\ \mathit{not}(m_1,w_2)\\
(m_1,w_4) &\leftarrow \mathit{not}(m_1,w_2),\ \mathit{not}(m_2,w_4)\\
(m_2,w_1) &\leftarrow \mathit{not}(m_2,w_3)\\
(m_2,w_2) &\leftarrow \mathit{not}(m_2,w_4),\ \mathit{not}(m_2,w_1),\ \mathit{not}(m_2,w_3),\ \mathit{not}(m_1,w_2),\ \mathit{not}(m_3,w_2),\\
&\qquad \mathit{not}(m_4,w_2)\\
(m_2,w_3) &\leftarrow \mathit{not}(m_3,w_3),\ \mathit{not}(m_4,w_3),\ \mathit{not}(m_1,w_3)\\
(m_2,w_4) &\leftarrow \mathit{not}(m_2,w_1),\ \mathit{not}(m_2,w_3),\\
(m_3,w_1) &\leftarrow \mathit{not}(m_3,w_3),\ \mathit{not}(m_3,w_2),\ \mathit{not}(m_4,w_1),\ \mathit{not}(m_1,w_1),\ \mathit{not}(m_1,w_2)\\
(m_3,w_2) &\leftarrow \mathit{not}(m_4,w_2)\\
(m_3,w_3) &\leftarrow \mathit{not}(m_3,w_2),\ \mathit{not}(m_4,w_3),\ \mathit{not}(m_1,w_3)\\
(m_3,w_4) &\leftarrow \mathit{not}(m_3,w_1),\ \mathit{not}(m_3,w_3),\ \mathit{not}(m_3,w_2),\ \mathit{not}(m_4,w_4),\ \mathit{not}(m_1,w_4),\\
&\qquad \mathit{not}(m_2,w_4)\\
(m_4,w_1) &\leftarrow \mathit{not}(m_4,w_4),\ \mathit{not}(m_1,w_1),\ \mathit{not}(m_2,w_1)\\
(m_4,w_2) &\leftarrow \mathit{not}(m_4,w_3),\ \mathit{not}(m_4,w_1),\ \mathit{not}(m_4,w_4)\\
(m_4,w_3) &\leftarrow \mathit{not}(m_4,w_1),\ \mathit{not}(m_4,w_4),\ \mathit{not}(m_1,w_3)\\
(m_4,w_4) &\leftarrow \mathit{not}(m_1,w_4),\ \mathit{not}(m_2,w_4)
\end{aligned}
$$

It can be checked that $\{(m_1,w_4), (m_2,w_3), (m_3,w_2), (m_4,w_1)\}$, and $\{(m_1,w_4), (m_2,w_1), (m_3,w_2), (m_4,w_3)\}$ are the answer sets of P, according to the Theorem 2.

We stress that the ASP modeling of the SMP introduced above is a direct and natural representation of the problem [20].

4 Stable Roommates Problem via Incoherent ASP Programs

In the literature, there exists several variants of the SMP [30, 32, 38, 34], those are also studied from a modeling perspective using ASP [19]. In the previous Section, we have pointed out that the sets M and W were disjoint. What happens if $M \cap W \neq \emptyset$? In particular, we can assume that the two sets coincide, we call A this set, and each person in A ranks all the others persons in order of preferences. This variant of the SMP is known as the Stable Roommates Problem (SRP) [22]. Clearly, to have a stable roommates a necessary condition is that the cardinality of A is even, otherwise no stable matching could exist.

More formally, let A be a set of $2n$ persons. For each person $p \in A$, we consider a bijective map $\pi_p : A \setminus \{p\} \to \{1,2,...,2n-1\}$ that assigns to each other person in A a number from 1 to $2n-1$, denoting the ranking. For instance, if $n=2$, we have a set of 4 persons $A = \{p_1, p_2, p_3, p_4\}$. Assume that person p_1 prefers person p_2 to person p_3, and prefers person p_3 to person p_4. Then, we will have that $\pi_{p_1}(p_2) = 1$, $\pi_{p_1}(p_3) = 2$, and $\pi_{p_1}(p_4) = 3$. The goal is to find a set S of n pairs, say A_1, A_2, ..., A_n that form a partition of A (i.e., $A_1 \cup A_2 \cup \ldots \cup A_n = A$, and $A_i \cap A_j = \emptyset$, for each $i \neq j$) such that no two persons who are not roommates both prefer each other to their actual partners.

Now, the SRP has a structure similar to the SMP. Hence, the modeling of the SMP via ASP can be applied as it is to the SRP. There is no substantial motivation to change it. It remains a direct and a natural encoding also for the SRP.

Theorem 3. *A set $S \subseteq A \times A$ constitutes a solution to the SRP if, and only if, S is an answer set of the corresponding ASP program.*

Example 4. Consider the following instance of the SRP: $A = \{p_1, p_2, p_3, p_4\}$, and

$$\begin{array}{ll} p_1 \text{ prefers } p_2 \text{ to } p_4; & p_1 \text{ prefers } p_4 \text{ to } p_3; \\ p_2 \text{ prefers } p_3 \text{ to } p_1; & p_2 \text{ prefers } p_1 \text{ to } p_4; \\ p_3 \text{ prefers } p_1 \text{ to } p_4; & p_3 \text{ prefers } p_4 \text{ to } p_2; \\ p_4 \text{ prefers } p_3 \text{ to } p_2; & p_4 \text{ prefers } p_2 \text{ to } p_1. \end{array}$$

So that, $\pi_{p_1}(p_2) = 1$, $\pi_{p_1}(p_3) = 3$, $\pi_{p_1}(p_4) = 2$, $\pi_{p_2}(p_1) = 2$, $\pi_{p_2}(p_3) = 1$, $\pi_{p_2}(p_4) = 3$, $\pi_{p_3}(p_1) = 1$, $\pi_{p_3}(p_2) = 3$, $\pi_{p_3}(p_4) = 2$, $\pi_{p_4}(p_1) = 3$, $\pi_{p_4}(p_2) = 2$, $\pi_{p_4}(p_3) = 1$. Therefore, the ASP program associated to this SRP instance is given by the following set of rules:

$$P = \left\{ \begin{array}{l} (p_1,p_2) \leftarrow not(p_2,p_3) \\ (p_1,p_3) \leftarrow not(p_1,p_4),\ not(p_1,p_2) \\ (p_1,p_4) \leftarrow not(p_1,p_2),\ not(p_2,p_4),\ not(p_3,p_4) \\ (p_2,p_3) \leftarrow not(p_3,p_4),\ not(p_1,p_3) \\ (p_2,p_4) \leftarrow not(p_1,p_2),\ not(p_2,p_3),\ not(p_3,p_4) \\ (p_3,p_4) \leftarrow not(p_1,p_3) \end{array} \right\}.$$

It can be checked that $\{(p_1, p_2), (p_3, p_4)\}$ is the unique solution to this SRP instance as well as the unique answer set of P, according to the Theorem 3.

However, unlike the SMP, a stable matching for the SRP may fail to exist for certain sets of persons and their preferences. In this case no answer set of the modeling program exists.

Example 5. Consider the following instance of the SRP: $A = \{p_1, p_2, p_3, p_4\}$, and

$$\begin{array}{ll} p_1 \text{ prefers } p_2 \text{ to } p_3; & p_1 \text{ prefers } p_3 \text{ to } p_4; \\ p_2 \text{ prefers } p_3 \text{ to } p_1; & p_2 \text{ prefers } p_1 \text{ to } p_4; \\ p_3 \text{ prefers } p_1 \text{ to } p_2; & p_3 \text{ prefers } p_2 \text{ to } p_4; \\ p_4 \text{ prefers } p_1 \text{ to } p_2; & p_4 \text{ prefers } p_2 \text{ to } p_3. \end{array}$$

So that, $\pi_{p_1}(p_2) = 1$, $\pi_{p_1}(p_3) = 2$, $\pi_{p_1}(p_4) = 3$, $\pi_{p_2}(p_1) = 2$, $\pi_{p_2}(p_3) = 1$, $\pi_{p_2}(p_4) = 3$, $\pi_{p_3}(p_1) = 1$, $\pi_{p_3}(p_2) = 2$, $\pi_{p_3}(p_4) = 3$, $\pi_{p_4}(p_1) = 1$, $\pi_{p_4}(p_2) = 2$, $\pi_{p_4}(p_3) = 3$. Therefore, the ASP program associated to this SRP instance is given by the following set of rules:

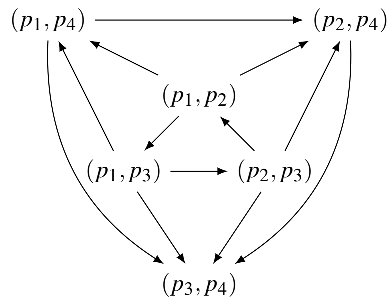

Fig. 1: Dependency graph of the program P of the Example 5.

$$P = \left\{ \begin{array}{l} (p_1, p_2) \leftarrow not(p_2, p_3) \\ (p_1, p_3) \leftarrow not(p_1, p_2) \\ (p_1, p_4) \leftarrow not(p_1, p_3),\ not(p_1, p_2) \\ (p_2, p_3) \leftarrow not(p_1, p_3) \\ (p_2, p_4) \leftarrow not(p_1, p_2),\ not(p_2, p_3),\ not(p_1, p_4) \\ (p_3, p_4) \leftarrow not(p_2, p_3),\ not(p_1, p_3),\ not(p_2, p_4),\ not(p_1, p_4) \end{array} \right\}.$$

To highlight the negative dependecies of each atom of the program, the dependency graph of P is reported in Figure 1. It can be checked that there is no solution to this SRP instance as well as no answer set of P exists, according to the Theorem 3.

Note that the lack of answer sets is not due to the violation of some constraint. Indeed, no constraint appears in the encoding above. But, it is caused by cyclic default negation. We point out that the absence of answer sets is not due to a wrong modeling approach. It concerns the intrinsic characteristics of the notion of answer set. However, it is noteworthy that in literature and, in particular, in all the benchmarks appearing in the ASP competitions, the lack of answer sets is always due to the violation of some constraint [26, 5, 6].

5 Conclusion

In this paper, we investigated a modeling of the SMP via ASP, by showing its naturalness through its relations with abstract argumentation modeling. However, the modeling proposed may lead to lack of answer sets due to cyclic default negation, when we move from the SMP to the SRP. Hence, this modeling approach to the SRP leads to have a first benchmark with consistent, but incoherent ASP programs.

References

1. Adrian, W.T., Manna, M., Leone, N., Amendola, G., Adrian, M.: Entity set expansion from the web via ASP. In: ICLP (Technical Communications). OASICS, vol. 58, pp. 1:1–1:5. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)

2. Alviano, M., Amendola, G., Peñaloza, R.: Minimal undefinedness for fuzzy answer sets. In: AAAI. pp. 3694–3700. AAAI Press (2017)
3. Amendola, G.: Dealing with incoherence in ASP: split semi-equilibrium semantics. In: DWAI@AI*IA. CEUR Workshop Proceedings, vol. 1334, pp. 23–32. CEUR-WS.org (2014)
4. Amendola, G.: Preliminary results on modeling interdependent scheduling games via answer set programming. In: RCRA@AI*IA. p. to appear. CEUR Workshop Proceedings, CEUR-WS.org (2018)
5. Amendola, G., Dodaro, C., Faber, W., Leone, N., Ricca, F.: On the computation of paracoherent answer sets. In: AAAI. pp. 1034–1040. AAAI Press (2017)
6. Amendola, G., Dodaro, C., Faber, W., Ricca, F.: Externally supported models for efficient computation of paracoherent answer sets. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, February 2-7, 2018, New Orleans, Louisiana, USA. pp. 1034–1040 (2018)
7. Amendola, G., Dodaro, C., Leone, N., Ricca, F.: On the application of answer set programming to the conference paper assignment problem. In: AI*IA. Lecture Notes in Computer Science, vol. 10037, pp. 164–178. Springer (2016)
8. Amendola, G., Dodaro, C., Ricca, F.: ASPQ: an asp-based 2qbf solver. In: QBF@SAT. CEUR Workshop Proceedings, vol. 1719, pp. 49–54. CEUR-WS.org (2016)
9. Amendola, G., Eiter, T., Fink, M., Leone, N., Moura, J.: Semi-equilibrium models for paracoherent answer set programs. Artif. Intell. **234**, 219–271 (2016)
10. Amendola, G., Eiter, T., Leone, N.: Modular paracoherent answer sets. In: Logics in Artificial Intelligence - 14th European Conference, JELIA2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings. pp. 457–471 (2014)
11. Amendola, G., Greco, G., Leone, N., Veltri, P.: Modeling and reasoning about NTU games via answer set programming. In: IJCAI 2016. pp. 38–45 (2016)
12. Amendola, G., Ricca, F., Truszczynski, M.: Generating hard random boolean formulas and disjunctive logic programs. In: IJCAI. pp. 532–538. ijcai.org (2017)
13. Amendola, G., Ricca, F., Truszczynski, M.: A generator of hard 2qbf formulas and asp programs. In: KR. AAAI Press (2018)
14. Amendola, G., Ricca, F., Truszczynski, M.: Random models of very hard 2qbf and disjunctive programs: An overview. In: ICTCS. CEUR Workshop Proceedings, CEUR-WS.org (2018)
15. Bonatti, P.A., Calimeri, F., Leone, N., Ricca, F.: Answer set programming. In: 25 Years GULP. Lecture Notes in Computer Science, vol. 6125, pp. 159–182. Springer (2010)
16. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Com. ACM **54**(12), 92–103 (2011)
17. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: Design and results of the Fifth Answer Set Programming Competition. Artif. Intell. **231**, 151–181 (2016)
18. Caminada, M., Sá, S., Alcântara, J., Dvorák, W.: On the equivalence between logic programming semantics and argumentation semantics. Int. J. Approx. Reasoning **58**, 87–111 (2015)
19. Clercq, S.D., Schockaert, S., Cock, M.D., Nowé, A.: Solving stable matching problems using answer set programming. TPLP **16**(3), 247–268 (2016)
20. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. **77**(2), 321–358 (1995)
21. Erdem, E., Öztok, U.: Generating explanations for biomedical queries. TPLP **15**(1), 35–78 (2015)
22. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. The American Mathematical Monthly **69**(1), 9–15 (1962), `http://www.jstor.org/stable/2312726`
23. Gebser, M., Leone, N., Maratea, M., Perri, S., Ricca, F., Schaub, T.: Evaluation techniques and systems for answer set programming: a survey. In: IJCAI 2018. pp. 5450–5456 (2018)

24. Gebser, M., Maratea, M., Ricca, F.: The Design of the Sixth Answer Set Programming Competition. In: LPNMR. LNCS, vol. 9345, pp. 531–544. Springer (2015)
25. Gebser, M., Maratea, M., Ricca, F.: What's hot in the answer set programming competition. In: AAAI. pp. 4327–4329. AAAI Press (2016)
26. Gebser, M., Maratea, M., Ricca, F.: The sixth answer set programming competition. Journal of Artificial Intelligence Research **60**, 41–95 (2017)
27. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. **9**(3/4), 365–386 (1991)
28. Grasso, G., Iiritano, S., Leone, N., Lio, V., Ricca, F., Scalise, F.: An asp-based system for team-building in the gioia-tauro seaport. In: PADL. Lecture Notes in Computer Science, vol. 5937, pp. 40–42. Springer (2010)
29. Grasso, G., Iiritano, S., Leone, N., Ricca, F.: Some DLV applications for knowledge management. In: LPNMR. Lecture Notes in Computer Science, vol. 5753, pp. 591–597. Springer (2009)
30. Gusfield, D.: Three fast algorithms for four problems in stable marriage. SIAM J. Comput. **16**(1), 111–128 (1987)
31. Gusfield, D., Irving, R.W.: The Stable marriage problem - structure and algorithms. Foundations of computing series, MIT Press (1989)
32. Irving, R.W., Leather, P., Gusfield, D.: An efficient algorithm for the "optimal" stable marriage. J. ACM **34**(3), 532–543 (1987)
33. Lierler, Y., Maratea, M., Ricca, F.: Systems, engineering environments, and competitions. AI Magazine **37**(3), 45–52 (2016)
34. Manlove, D., Irving, R.W., Iwama, K., Miyazaki, S., Morita, Y.: Hard variants of stable marriage. Theor. Comput. Sci. **276**(1-2), 261–279 (2002)
35. Manna, M., Ricca, F., Terracina, G.: Taming primary key violations to query large inconsistent data via ASP. TPLP **15**(4-5), 696–710 (2015)
36. Maratea, M., Pulina, L., Ricca, F.: A multi-engine approach to answer-set programming. TPLP **14**(6), 841–868 (2014)
37. Maratea, M., Ricca, F., Faber, W., Leone, N.: Look-back techniques and heuristics in DLV: implementation, evaluation, and comparison to QBF solvers. J. Algorithms **63**(1-3), 70–89 (2008)
38. McDermid, E., Irving, R.W.: Sex-equal stable matchings: Complexity and exact algorithms. Algorithmica **68**(3), 545–570 (2014)
39. Wu, Y., Caminada, M., Gabbay, D.M.: Complete extensions in argumentation coincide with 3-valued stable models in logic programming. Studia Logica **93**(2-3), 383–403 (2009)