August, 2020

Basics behind Answer Sets

Yuliya Lierler

Handout on Basics behind Answer Sets

Yuliya Lierler
University of Nebraska Omaha

Introduction

Answer set programming (ASP) is a form of declarative programming oriented towards modeling

i intelligent agents in knowledge representation and reasoning and

ii difficult combinatorial search problems.

It belongs to the group of so called constraint programming languages. ASP has been applied to a variety of applications including plan generation and product configuration problems in artificial intelligence and graph-theoretic problems arising in VLSI design and in historical linguistics [1].

Syntactically, ASP programs look like logic programs in Prolog, but the computational mechanisms used in ASP are different: they are based on the ideas stemming from the development of satisfiability solvers for propositional logic.

This document presents the concept of an answer set for programs in their simplest form: *no variables, no classical negation symbol, no disjunction in the heads of rules.* The textbooks [**?**, 2] provide an account for general definition of this concept that assumes rules of the general form that includes all the features mentioned above. Yet, it is useful to first consider as simple programs as possible to build intuitions about answer sets.

In this note italics is primarily used to identify concepts that are being defined. Some definitions are identified by the word **Definition**.

Traditional Programs and their Answer Sets

1 Syntax

A *traditional rule* is an expression of the form

$$a_0 \leftarrow a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n. \tag{1}$$

where $n \geq m \geq 0$; a_0 is a propositional atom or symbol $\perp$; and $a_1, \ldots, a_n$ are propositional atoms (propositional symbols). The expression a_0 is called the *head* of the rule, and the list

$$a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n$$

is its *body*. If the body is empty ($n = 0$) then the rule is called a *fact* and identified with its head a_0. We call rule (1) a *constraint* when its head is symbol $\perp$ (we then drop $\perp$ from a rule).

A *traditional (or propositional logic) program* is a finite set of traditional rules. For instance,

$$\begin{array}{l} p. \\ r \leftarrow p, q. \end{array} \tag{2}$$

and

$$\begin{array}{l} p \leftarrow not\ q. \\ q \leftarrow not\ r. \end{array} \tag{3}$$

are traditional programs.

A traditional rule (1) is *positive* if $m = n$, that is to say, if it has the form

$$a_0 \leftarrow a_1, \ldots, a_m. \tag{4}$$

A traditional program is *positive* if each of its rules is positive. For instance, program (2) is positive, and (3) is not.

2 The Answer Set of a Positive Program

We will first define the concept of an answer set for positive traditional programs. To begin, we introduce auxiliary definitions.

Definition 1. *A set X of atoms* satisfies *a positive traditional rule* (4) *when $a_0 \in X$ whenever $\{a_1, \ldots, a_m\} \subseteq X$.*

For instance, any positive traditional rule (4) is satisfied by a singleton set $\{a_0\}$.

To interpret Definition 1 recall the truth table of implication in propositional logic:

p	q	$p \rightarrow q$
$true$	$true$	$true$
$true$	$false$	$false$
$false$	$true$	$true$
$false$	$false$	$true$

One can intuitively read it in English as follows condition $p \to q$ holds if q holds whenever p holds. Expression $a_0 \in X$ plays a role of q whereas $\{a_1, \ldots, a_m\} \subseteq X$ plays a role of p in the definition of a set of atoms satisfying a rule.

Problem 1. *Given a set X of atoms and a positive traditional rule* (4)

	Does X satisfies rule (4)*?*
$\{a_1, \ldots, a_m\} \subseteq X$ *and* $a_0 \in X$	*Yes*
$\{a_1, \ldots, a_m\} \subseteq X$ *and* $a_0 \notin X$	
$\{a_1, \ldots, a_m\} \not\subseteq X$ *and* $a_0 \in X$	*Yes*
$\{a_1, \ldots, a_m\} \not\subseteq X$ *and* $a_0 \notin X$	

Definition 2. *A set X of atoms* satisfies *a positive traditional program* Π *if X satisfies every rule (4) in* Π.

For instance, any positive traditional program is satisfied by the set composed of the heads a_0 of all its rules (4).

Problem 2.

X	*Does X satisfies program* (2)*?*
$\emptyset$	*No*
$\{p\}$	*Yes*
$\{q\}$	
$\{r\}$	
$\{p\ q\}$	
$\{p\ r\}$	
$\{q\ r\}$	
$\{p\ q\ r\}$	

Proposition 1. *For any positive traditional program* Π *without constraints, there exists a set of atoms satisfying* Π.

Proof. Indeed. Consider the set X to be composed of all atoms occurring in Π. It follows that for every rule in Π its head atom is in X. By the definition of rule satisfaction it follows that X satisfies every rule of Π. (Note that we could have also considered other sets to construct a similar argument. Can you think of such a set?) □

Proposition 2. *For any positive traditional program* Π, *the intersection of all sets satisfying* Π *satisfies* Π *also.*

Proof. By contradiction. Suppose that this is not the case. Let X denote the intersection of all sets satisfying Π. By definition (of program's satisfiability), there exists a rule

$$a_0 \leftarrow a_1, \ldots, a_m.$$

in Π such that it is not satisfied by X, in other words

$$a_0 \notin X$$

and

$$\{a_1, \ldots, a_m\} \subseteq X.$$

Since X is an intersection of all sets satisfying Π then we conclude that (i) $\{a_1, \ldots, a_m\}$ belongs to each one of the sets satisfying Π and (ii) there is a set Y satisfying Π such that $a_0 \notin Y$. By definition, Y does not satisfy Π. We derive at contradiction.

$\square$

Proposition 2 allows us to talk about the *smallest* set of atoms that satisfies Π.

Definition 3. *The smallest set of atoms that satisfies positive traditional program* Π *is called* the answer set *of* Π*.*

For instance, the sets of atoms satisfying program (2) are

$$\{p\},\ \{p, r\},\ \{p, q, r\},$$

and its answer set is $\{p\}$.

We now illustrate some interesting properties of answer sets. Let a program consist of facts only. The set of these facts form the only answer set of such a program. Intuitively, each fact states what is known and an answer set reflects this information by asserting that each atom mentioned as a fact is *true*, whereas anything else is *false*. Thus answer sets semantics follows closed world assumption (CWA) – presumption that what is not currently known to be *true* is *false*. From the definition of an answer set and Proposition 1, it immediately follows that any positive traditional program has a unique answer set. It is intuitive to argue that answer set semantics for logic programs generalizes CWA. Note that an atom, which does not occur in the head of some rule in a program, will not be a part of any answer set of this program:

Proposition 3. *If* X *is an answer set of a positive traditional program* Π*, then every element of* X *is the head of one of the rules of* Π*.*

Positive Rules Intuitively, we can think of (4) when its head is an atom as a rule for generating atoms: once you have generated $a_1, \ldots, a_m$, you are allowed to generate a_0. The answer set is the set of all atoms that can be generated by applying rules of the program in any order. For instance, the first rule of (2) allows us to include p in the answer set. The second rule says that we can add r to the answer set if we have already included p and q. Given these two rules only, we can generate no atoms besides p. If we extend program (2) by adding the rule

$$q \leftarrow p.$$

then the answer set will become $\{p, q, r\}$. We can easily implement such a process for generating the answer set for positive traditional program by an efficient procedure.

Positive rules may remind you *Horn clauses* or *definite clauses*. One can identify (4) with the following implication

$$a_1 \wedge \cdots \wedge a_m \Rightarrow a_0$$

that is equivalent to the Horn clause

$$\neg a_1 \vee \cdots \vee \neg a_m \vee a_0.$$

Rule (4) is satisfied by a set of atoms if and only if its respective Horn clause is satisfied by this set in propositional logic.

3 Answer Sets of a Program with Negation

To extend the definition of an answer set to arbitrary traditional programs, we will introduce one more auxiliary definition.

Definition 4. *The* reduct Π^X *of a traditional program* Π *relative to a set* X *of atoms is the set of rules (4) for all rules (1) in* Π *such that* $a_{m+1}, \ldots, a_n \notin X$.

In other words, Π^X is constructed from Π by (i) dropping all rules (1) such that at least one atom from $a_{m+1}, \ldots, a_n$ is in X, and (ii) eliminating *not* $a_{m+1}, \ldots,$ *not* a_n expression from the rest of the rules.

Thus Π^X is a positive traditional program.

Problem 3. *Let* Π *be* (3),

X	*What is* Π^X*?*	*Explanation*
$\emptyset$	$p.$ $q.$	$p \leftarrow \cancel{not\ q.}$ $q \leftarrow \cancel{not\ r.}$
$\{p\}$	$p.$ $q.$	$p \leftarrow \cancel{not\ q.}$ $q \leftarrow \cancel{not\ r.}$
$\{q\}$	$q.$	$\cancel{p \leftarrow not\ q.}$ $q \leftarrow \cancel{not\ r.}$
$\{r\}$		
$\{p\ q\}$		
$\{p\ r\}$		
$\{q\ r\}$		
$\{p\ q\ r\}$		

Definition 5. *We say that* X *is an* answer set *of* Π *if* X *is the answer set of* Π^X *(that is, the smallest set of atoms satisfying* Π^X*).*

Problem 4.

X	*Is* X *an answer set of program* (3)*?*
$\emptyset$	*No*
$\{p\}$	*No*
$\{q\}$	*Yes*
$\{r\}$	
$\{p\ q\}$	
$\{p\ r\}$	
$\{q\ r\}$	
$\{p\ q\ r\}$	

If Π is positive then, for any X, $\Pi^X = \Pi$. It follows that the new definition of an answer set is a generalization of the definition from Section 2: for any positive traditional program Π, X is the smallest set of atoms satisfying Π^X iff X is the smallest set of atoms satisfying Π.

Intuitively, rule (1) allows us to generate a_0 as soon as we generated the atoms $a_1, \dots, a_m$ *provided that none of the atoms* $a_{m+1}, \dots, a_n$ *can be* *generated using the rules of the program.* There is a vicious circle in this sentence: to decide whether a rule of Π can be used to generate a new atom, we need to know which atoms can be generated using the rules of Π. The definition of an answer set overcomes this difficulty by employing a "fixpoint construction." Take a set X that you suspect may be exactly the set of atoms that can be generated using the rules of Π. Under this assumption, Π has the same meaning as the positive program Π^X. Consider the answer set of Π^X, as defined in Section 2. If this set is exactly identical to the set X that you started with then X was a "good guess"; it is indeed an answer set of Π.

In Problem 4, to find all answer sets of program (3) we constructed its reduct for each subset of $\{p, q, r\}$ to establish whether these sets are answer sets of (3). The following general properties of answer sets of traditional programs allow us to sometime establish that a set is not an answer set in a trivial way by inspecting its elements rather than constructing the reduct of a given program.

Proposition 4. *If X is an answer set of a traditional program Π then every element of X is the head of one of the rules of Π.*

Proposition 5. *If X is an answer set for a traditional program Π then no proper subset of X can be an answer set of Π.*

In application to program (3), Proposition 4 tells us that its answer sets do not contain r, so that we only need to check

$$\emptyset, \{p\}, \{q\}, \text{ and } \{p, q\}.$$

Once we established that $\{q\}$ is an answer set, by Proposition 5

- $\emptyset$ cannot be an answer set because it is a proper subset of the answer set $\{q\}$, and
- $\{p, q\}$ cannot be an answer set because the answer set $\{q\}$ is its proper subset.

Set $\{p\}$ is not an answer set as set $\{q\}$ is the answer set of program's reduct wrt $\{p\}$. Consequently, $\{q\}$ is the only answer set of (3).

Program (3) has a unique answer set. On the other hand, the program

$$\begin{array}{l} p \leftarrow not\ q. \\ q \leftarrow not\ p. \end{array} \tag{5}$$

has two answer sets: $\{p\}$ and $\{q\}$. The one-rule program

$$r \leftarrow not\ r. \tag{6}$$

has no answer sets.

Problem 5. *Prove that if X is an answer set of a traditional program Π so that for some rule* (1)*, it holds that $\{a_1, \ldots, a_m\} \subseteq X$ and $\{a_{m+1}, \ldots, a_n\} \cap X = \emptyset$, then $a_0 \in X$.*

Problem 6. *Find all answer sets of the following program, which extends (5) by two additional rules:*

$$\begin{array}{l} p \leftarrow not\ q. \\ q \leftarrow not\ p. \\ r \leftarrow p. \\ r \leftarrow q. \end{array}$$

Problem 7. *Find all answer sets of the following combination of programs (5) and (6):*

$$\begin{array}{l} p \leftarrow not\ q. \\ q \leftarrow not\ p. \\ r \leftarrow not\ r. \\ r \leftarrow p. \end{array}$$

Problem 8. *Prove the claim of Proposition 4*

Constraints Consider a constraint

$$\leftarrow p.$$

Extending program (2) by this rule will result in a program that has no answer sets. In other words, constraint $\leftarrow p$ eliminates the only answer of (2). It is convenient to view any constraint

$$\leftarrow a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n$$

as a clause (a disjunction of literals)

$$\neg a_1 \vee \cdots \vee \neg a_m \vee a_{m+1} \vee \cdots \vee a_n.$$

Then, we can state the general property about constraints: answer sets of a program satisfy the propositional logic formula composed of its constraints (here (i) the notion of satisfaction is as understood classically in propositional logic and (ii) an answer set is associated with an interpretation in an intuitive manner). Furthermore, for a program Π and a set Γ of constraints the answer sets of $\Pi \cup \Gamma$ coincide with the answer sets of Π that satisfy Γ. Consequently, constraints can be seen as elements of classical logic in logic programs.

4 Classical Negation in Programs

The textbook [2] immediately introduces programs that are of more complex form even in propositional case. Indeed, it allows additional connective $\neg$ (classical negation). So that basic entities of a program are literals (a *literal* is either an atom a or an atom proceeded with classical negation $\neg a$) In turn, the concept of an answer set is defined over the consistent sets of literals whereas here we defined an answer set over the sets of atoms. Yet, it is easy to *simulate* classical negation in the simpler form of programs that we consider here.

Classical negation can always be eliminated from a program by means of auxiliary atoms and additional constraints. Indeed, given a program composed of rules of the form

$$l_0 \leftarrow l_1, \dots, l_m, not\ l_{m+1}, \dots, not\ l_n \tag{7}$$

so that l_0 is a literal or $\bot$ and $l_1 \dots l_n$ are literals, we can replace an occurrence of any literal of the form $\neg a$ with a fresh atom a' and add a constraint to this program in the form

$$\leftarrow a, a'.$$

The answer sets of this new program as defined in this handout are in one to one correspondence with the answer sets as defined in the textbook. In particular, by replacing atoms of the form a' by $\neg a$ we obtain the textbook answer sets.

Acknowledgments

Parts of these notes follow the lecture notes on *Answer Sets; and Methodology of Answer Set Programming; course Answer set programming: CS395T, Spring 2005*[1] by Vladimir Lifschitz. Zachary Hansen helped to correct some typos.

[1] `http://www.cs.utexas.edu/~vl/teaching/asp.html`

References

[1] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[2] Michael Gelfond and Yulia Kahl. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press, 2014.