A primer on Answer Set Programming *

Alessandro Provetti

Dept. of Physics, Univ. of Messina, Italy.
ale@unime.it
http://mag.dsi.unimi.it/

Syntax

The following definitions describe the language DATALOG$^{\neg}$ as well as logic programs with no function symbols.
Assume a language of constants and predicate constants. Assume also that terms and atoms are built as in the corresponding first-order language. Unlike classical logic and standard logic programming, no function symbols are allowed. A rule is an expression of the form:

$$\rho \,:\, A_0 \leftarrow A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n \tag{1}$$

where $A_0, \ldots A_n$ are atoms and *not* is a logical connective called *negation as failure*. Also, for every rule let us define $head(\rho) = A_0$, $pos(\rho) = A_1, \ldots, A_m$, $neg(\rho) = A_{m+1}, \ldots, A_n$ and $body(\rho) = pos(\rho) \cup neg(\rho)$. The head of rules is never empty, while if $body(\rho) = \emptyset$ we refer to ρ as *a fact*.

A logic program is defined as a collection of rules. Rules with variables are taken as shorthand for the sets of all their ground instantiations and the set of all ground atoms in the language of a program Π will be denoted by $\mathbb{B}_{\Pi}$.
Queries and constraints are expressions with the same structure of rules but with empty head.

Semantics

Intuitively, a *stable model,* also called *answer set,* is a possible view of the world that is *compatible* with the rules of the program. Rules are therefore seen as constraints on these views of the world.

* Several portions of this document reproduce definitions given in [GelLif88] and elsewhere. This work was supported by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2001-37004 WASP project.

Let us start defining stable models/answer sets of the subclass of positive programs, i.e. those where, for every rule ρ, $neg(\rho) = \emptyset$.

Definition 1. *(Stable model of positive programs)*

The stable model $a(\Pi)$ *of a positive program* Π *is the smallest subset of* B_Π *such that for any rule (1) in* Π*:*

$$A_1, \ldots, A_m \in a(\Pi) \Rightarrow A_0 \in a(\Pi) \tag{2}$$

Clearly, positive programs have a unique stable model, which coincides with that obtained applying other semantics; in other words positive programs are unambiguous. Moreover, the stable model of positive programs can be obtained as the fixpoint of the *immediate consequence operator* T_Π iterated from $\emptyset$ on.

Definition 2. *(Stable models of programs)*

Let Π *be a logic program. For any set* S *of atoms, let* $\Gamma(\Pi, S)$ *be a program obtained from* Π *by deleting*

(i) *each rule that has a formula "not* A*" in its body with* $A \in S$*;*

(ii) *all formulae of the form "not* A*" in the bodies of the remaining rules.*

Clearly, $\Gamma(\Pi, S)$ *does not contain not , so that its stable model is already defined. If this stable model coincides with* S*, then we say that* S *is a* stable model *of* Π*. In other words, a stable model of* Π *is characterized by the equation:*

$$S = a(\Gamma(\Pi, S)). \tag{3}$$

Programs which have a unique stable model are called categorical.

Let us define entailment in the stable models semantics. A ground atom α is *true in* S if $\alpha \in S$, otherwise α is *false*, i.e., by abuse of notation, $\neg\alpha$ is *true* is S. This definition can extended to arbitrary first-order formulae in the standard way.

We will say that Π *entails a formula* ϕ (written $\Pi \models \phi$) if ϕ is true in *all* the stable models of Π. We will say that the answer to a ground query γ is *yes* if γ is true in all stable models of Π, i.e. $\Pi \models \gamma$;

no if $\neg\gamma$ is true in all stable models of Π, i.e. $\Pi \models \neg\gamma$;

unknown otherwise.

It is easy to see that logic programs are *nonmonotonic*, i.e. adding new information to the program may force a reasoner associated with it to withdraw its previous conclusions.

Corollary 1. *(Gelfond and Lifschitz [GelLif91])*
If an extended logic program has an inconsistent Answer set, this is unique.

□

For programs without explicit negation stable models and answer sets coincide, so that in the following we will refer to [consistent]answer sets or stable models indifferently.

1 Reasoning with Answer Sets

In the following we report a basic result from Marek and Subramanian which -together with its corollaries- will be used in proofs about logic programs.

The result is slightly more general than the original, as it refers to answer sets and it is given a simple proof based on minimality.

Lemma 1 (Marek and Subramanian). *The following result on answer sets is due to Marek and Subramanian, originally for general logic programs.*

For any answer set A of an extended logic program Π:

– *For any ground instance of a rule of the type:*

$$L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n \quad (4)$$

from Π, if

$$\{L_1, \ldots, L_m\} \subseteq A \ and\ \{L_{m+1}, \ldots, L_n\} \cap A = 0$$

then $L_0 \in A$.

– *If A is a consistent Answer set of Π and $L_0 \in A$, then there exists a ground instance rule of type 4 from Π such that:*

$$\{L_1, \ldots, L_m\} \subseteq A \text{ and } \{L_{m+1}, \ldots, L_n\} \cap A = 0.$$

□

Corollary 2. *If $\{L \leftarrow\} \in \Pi$ then L belongs to every Answer set of Π. It follows directly from Lemma 1.*

□

Definition 3. *We will say that an axiom r* supports *a literal L if the head of r matches with L. Moreover, we say that L is* supported only by *r if there is no other ground rule whith head L.*

Definition 4. *We will say that a rule r* justifies a literal L w.r.t. an answer set *A if*

a) r supports L;

b) r satisfies the conditions set forth in the first half of Lemma 1 w.r.t. A: the atoms occurring positively in the body being in A while those occurring negatively being not.

Clearly, justified literals belong to A.

Corollary 3. *If A is a consistent answer set of Π, $L_0 \in A$, and L_0 is supported only by an axiom r of type (4) from Π then:*

$$\{L_1, \ldots, L_m\} \subseteq A \text{ and } \{L_{m+1}, \ldots, L_n\} \cap A = 0.$$

It follows directly from Lemma 1.

□

2 Examples

Example 1. $\pi_1 =$
happy $\leftarrow$ *not sad.*
sad $\leftarrow$ *not happy.*

has two answer sets: $\{happy\}$ and $\{sad\}$.

Example 2. $\pi_2 =$
$happy \leftarrow not\ sad.$
$sad \leftarrow not\ soandso.$
$soandso \leftarrow not\ happy.$

has no answer set.

Example 3. $\pi_3 =$
$drinks \leftarrow happy.$
$drinks \leftarrow sad.$
$happy \leftarrow not\ sad.$
$sad \leftarrow not\ happy.$

has two answer sets: $\{drinks, happy\}$ and $\{drinks, sad\}$.

Example 4. $\pi_4 =$
$soandso \leftarrow not\ sad, not\ happy.$
$happy \leftarrow not\ sad, not\ soandso.$
$sad \leftarrow not\ happy, not\ soandso.$

has three answer sets: $\{happy\}$ and $\{sad\}$ and $\{soandso\}$.

Example 5. $\pi_5 =$
$f \leftarrow not\ f, not\ a.$
$a \leftarrow not\ b.$
$b \leftarrow not\ a.$

has only one answer set: $\{a\}$.

Example 6. $\pi_6 =$
$f \leftarrow not\ f, a.$
$a \leftarrow not\ b.$
$b \leftarrow not\ a.$

has only one answer set: $\{b\}$.

Exercise 1. $\pi_x =$
$f \leftarrow b.$
$c \leftarrow a.$
$a \leftarrow d.$
$d \leftarrow not\ b.$
$b \leftarrow not\ a.$

2.1 Examples with explicit negation

Example 7. $\pi_7 =$
$\neg a \leftarrow not\ a.$
$b \leftarrow \neg a.$

has only one answer set: $\{b, \neg a\}$.

3 Sources

Several ASP solvers are now available and can be downloaded from [Solvers].
A textbook on Answer Set Programming is now available [Bar03], and exercises can be downloaded from there.

References

[AptBol94] Apt K. R. and Bol R.N.. 1994. *Logic programming and negation: a survey.* Journal of Logic Programming, 19/20:9–71.

[BarGel94] Baral, C. and Gelfond. M., 1994. *Logic programming and knowledge representation,* J. of Logic Programming, 19/20:73–148.

[Bar03] Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving* Cambridge University Press. *http://www.baral.us/bookone/*

[GelLif88] Gelfond, M. and Lifschitz, V., 1988. *The stable model semantics for logic programming,* Proc. of 5th ILPS conference, pp. 1070–1080.

[GelLif91] M. Gelfond and V. Lifschitz., 1991. *Classical negation in logic programs and disjunctive databases.* New Generation Computing, 9(3/4):365–386.

[MarTru99] W. Marek, and M. Truszczyński. *Stable models and an alternative logic programming paradigm,* The Logic Programming Paradigm: a 25-Year Perspective, Springer-Verlag, 75–398. CoRR cs.LO/9809032.

[Solvers] Web location of some ASP solvers:
Cmodels: *http://www.cs.utexas.edu/users/tag/cmodels.html*
DLV: *http://www.dbai.tuwien.ac.at/proj/dlv/*
NoMoRe: *http://www.cs.uni-potsdam.de/~linke/nomore/*
SMODELS: *http://www.tcs.hut.fi/Software/smodels/*

[SacZan97] Saccà D. and Zaniolo C., 1997. *Deterministic and Non-Deterministic Stable Models.* J. of Logic and Computation 7(5):555–579.