# A primer on Answer Set Programming*

Department of Computer Science
The University of Texas at El Paso

## 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:

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

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

* By Ale Provetti, *provetti@cs.utep.edu*, 915-747-6956.

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

*The* stable model $a(\Pi)$ *of a positive program* $\Pi$ *is the smallest subset of* $\mathrm{B}_\Pi$ *such that for any rule (1) in* $\Pi$*:*

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

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* $T_\Pi$ iterated from $\emptyset$ on.

**Definition 2.** *(Stable models of programs)*

*Let* $\Pi$ *be a logic program. For any set* $S$ *of atoms, let* $\Gamma(\Pi, S)$ *be a program obtained from* $\Pi$ *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* $\Pi$*. In other words, a stable model of* $\Pi$ *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 $\alpha$ is *true in* $S$ if $\alpha \in S$, otherwise $\alpha$ 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 $\Pi$ *entails a formula* $\phi$ ( written $\Pi \models \phi$) if $\phi$ is true in *all* the stable models of $\Pi$. We will say that the answer to a ground query $\gamma$ is

*yes* if $\gamma$ is true in all stable models of $\Pi$, i.e. $\Pi \models \gamma$;

*no* if $\neg\gamma$ is true in all stable models of $\Pi$, 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.

## Programs with Explicit negation

Starting from Stable models semantics, Gelfond and Lifschitz have introduced the class of extended programs, i.e. those were atoms may appear with explicit negation in front:

$$A/\neg A$$

Atoms and their explicit negations are called literals. For extended programs the answer sets semantics is defined, which basically resembles stable models but if an answer set contains both $A$ and $\neg A$ then it contains all literals (ex contraditione quod libet sequitur).

**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 $\Pi$:*

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

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

  *from $\Pi$, if*

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

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

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

□

## 2 Examples

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

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

*Example 2.* $\pi_2 =$
*happy* ← *not sad.*
*sad* ← *not soandso.*
*soandso* ← *not happy.*

has no answer set.

*Example 3.* $\pi_3 =$
*drinks* ← *happy.*
*drinks* ← *sad.*
*happy* ← *not sad.*
*sad* ← *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.

[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, pp. 365–387.

[MarTru99] W. Marek, and M. Truszczyński. *Stable models and an alternative logic programming paradigm,* The Logic Programming Paradigm: a 25-Year Perspective, Springer-Verlag: 375–398.

[NieSim98] Niemelä I. and Simons P., 1998. *Logic programs with stable model semantics as a constraint programming paradigm.* Proc. of NM'98 workshop. Extended version submitted for publication.

[Solvers] Web location of some ASP solvers:
CCALC: *http://www.cs.utexas.edu/users/mcain/cc*
Cmodels: *http://www.cs.utexas.edu/users/tag/cmodels.html*
DeReS: *http://www.cs.engr.uky.edu/~lpnmr/DeReS.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.