

# **ASet-Prolog: A-Prolog with Sets and Aggregates**

Presented By

by

Veena S. Mellarkod

August 21, 2003

# Structure

The talk is structured as follows:

- Language
- Implementation Details
- Future Work

# ASet-Prolog

- ASet-Prolog is an extension of A-Prolog by sets and functions from sets to natural numbers (aggregates).
- There are 3 kinds of atoms in the language:  
*r – atoms*, *s – atoms* and *f – atoms*.

# Atoms

**r-atoms:** regular atoms of A-Prolog.

**s-atoms:** expressions of the form,

$$\{\bar{X} : p(\bar{X})\} \subseteq \{\bar{X} : q(\bar{X})\}$$

**f-atoms:** expressions of the form,

$$t = f(\{\bar{X} : p(\bar{X})\})$$

$f$  can be card, sum, max, min etc.,

# Programs

An ASet-Prolog program is a collection of rules of the form:

$$l_0 :- l_1, \dots, l_m, \textit{not } l_{m+1}, \dots, \textit{not } l_n.$$

where  $l_1, \dots, l_n$  are arbitrary atoms and  $l_0$  is either an *r - atom* or *s - atom* in ASet-Prolog.

Example:

$$\Pi_0 \left\{ \begin{array}{l} p(a). \quad p(b). \\ \{X : q(X)\} \subseteq \{X : p(X)\}. \end{array} \right.$$

Answer Sets:

{ p(a), p(b) }

{ p(a), p(b), q(a) }

{ p(a), p(b), q(b) }

{ p(a), p(b), q(a), q(b) }

## Examples

$$\Pi_1 \left\{ \begin{array}{l} p(a). \quad p(b). \\ \{X : q(X)\} \subseteq \{X : p(X)\}. \\ :- \quad T = \text{card}(\{X : q(X)\}), T \leq 1. \end{array} \right.$$

Answer Set:  $\{ p(a), p(b), q(a), q(b) \}$

$$\Pi_2 \left\{ \begin{array}{l} p(a). \quad p(b). \quad r(a). \\ s(a) :- \{X : r(X)\} \subseteq \{X : p(X)\}. \end{array} \right.$$

Answer Set:  $\{ p(a), p(b), r(a), s(a) \}$

# Semantics

Let  $S$  be a set of ground  $r$ -atoms in  $\Sigma_{\square}$ .

1. An  $r$  – atom  $l$ , in  $\Sigma_{\square}$  is true in  $S$ , if  $l \in S$
2. An  $s$  – atom is true in  $S$ , if for any sequence  $\bar{x}$  of ground terms of  $\Sigma_{\square}$ , either
  - $p(\bar{x}) \notin S$  or
  - $q(\bar{x}) \in S$ .
3. An  $f$  – atom is true in  $S$ , if value of the aggregate of the set  $\{\bar{x} : p(\bar{x}) \in S\}$  is equal to  $t$ .

## Answer Sets of ASet

Let  $S$  be a collection of ground  $r - atoms$ .  
 $se(\Pi, S)$  is the program obtained by:

1. removing from  $\Pi$  all the rules whose bodies contain  $s - atoms$  or  $f - atoms$  not satisfied by  $S$ ;
2. removing all remaining  $s - atoms$  and  $f - atoms$  from the bodies of the rules;
3. replacing rules of the form  $l \leftarrow \Gamma$  where  $l$  is an  $s - atom$  not satisfied by  $S$  by rules  $\leftarrow \Gamma$ ;
4. replacing the remaining rules of the form:  
 $\{\bar{x} : p(\bar{x})\} \subseteq \{\bar{x} : q(\bar{x})\} \leftarrow \Gamma$  by the rules  
 $p(\bar{t}) \leftarrow \Gamma$  for each  $p(\bar{t})$  from  $S$ .

$S$  is an answer set of  $\Pi$  if it is an answer set of  $se(\Pi, S)$ .

## Example

$$\Pi_3 \left\{ \begin{array}{l} p(1). p(2). r(1). \\ q(a) :- \{X : p(X)\} \subseteq \{X : r(X)\}. \\ q(b) :- \{X : r(X)\} \subseteq \{X : p(X)\}. \\ q(c) :- T = \text{card}(\{X : r(X)\}), T > 1. \\ q(d) :- T = \text{sum}(\{X : r(X)\}), T = 1. \\ \{X : r(X)\} \subseteq \{X : p(X)\}. \end{array} \right.$$

## Example Contd.

Let  $S = \{p(1), p(2), r(1), q(b), q(d)\}$  then  $se(\Pi_0, S)$  is:

$p(1). p(2). r(1).$

$q(b) :-$

$q(d) :-$

$r(1) :-$

$S$  is a stable model of  $se(\Pi_0, S)$ .

## Example Contd.

Let  $S_1 = \{p(1), p(2), r(1), r(2), q(a), q(b), q(c)\}$   
then  $se(\Pi_0, S_1)$  is:

$p(1). p(2). r(1).$

$q(a) :-$

$q(b) :-$

$q(c) :-$

$r(1) :-$

$r(2) :-$

$S_1$  is a stable model of  $se(\Pi_0, S_1)$ .

## Example Contd.

Let  $S_2 = \{p(1), p(2), r(1), q(a), q(c)\}$  then  $se(\Pi_0, S_2)$  is:

$p(1). p(2). r(1).$

$q(b) :-$

$q(d) :-$

$r(1) :-$

$S_2$  is not a stable model of  $se(\Pi_0, S_2)$ ..

# Coloring of Graphs

Given a graph  $G$  defined by a set of facts of the form  $node(X)$  and  $edge(X,Y)$  and a set  $C$  of colors  $color(red)$ ,  $color(green)$  etc., The coloring problem can be represented by a program  $\Pi$ :

$$\{C : colored(X, C)\} \subseteq \{C : color(C)\} :- node(X).$$

$$:- N = card(\{C : colored(X, C)\}), N \neq 1.$$

$$:- colored(X, C), colored(Y, C), edge(X, Y).$$

# Course Pre-requisites

Given a record of courses passed by a student  $s$ , as facts:  $passed(s,c1)$ ,  $passed(s, c2)$  and  $passed(s, c3)$  and a list of pre-requisites for each class as facts:  $prereq(c1,c4)$ ,  $prereq(c2,c4)$ , and  $pre-req(c4,c5)$ , the rule that a student  $S$  is allowed to take class  $C$  if he passed all the pre-requisites for  $C$  and didn't pass  $C$  yet, can be written as:

$$\begin{aligned} &can\_take(S, C) :- \\ &\{X : prereq(X, C)\} \subseteq \{X : passed(S, X)\}, \\ &not\ passed(S, C). \end{aligned}$$

# Sets and Choice Rules

An Smodels Program:

$$\begin{aligned} & d_1(1). \\ & d_2(1). \quad d_2(2). \\ & \{p(X) : d_1(X)\}. \\ & \{p(X) : d_2(X)\}. \end{aligned}$$

Stable Models of the Program:

$$\begin{aligned} & \{ \} \\ & \{p(1)\} \\ & \{p(2)\} \\ & \{p(1), p(2)\} \end{aligned}$$

# Sets and Choice Rules

An ASET-Prolog Program:

$$\begin{aligned} & d_1(1). \\ & d_2(1). \quad d_2(2). \\ & \{X : p(X)\} \subseteq \{X : d_1(X)\}. \\ & \{X : p(X)\} \subseteq \{X : d_2(X)\}. \end{aligned}$$

Answer sets of the Program:

$$\begin{aligned} & \{ \} \\ & \{p(1)\} \end{aligned}$$

# Difference between Sets and Choice Rules

Example:

$$\sqcap \left\{ \begin{array}{l} d_1(1). d_2(2). \\ a. \\ \{X : p(X)\} \subseteq \{X : d_1(X)\} :- a. \\ p(X) :- d_2(X). \end{array} \right.$$

ASet Returns: No Answer Set.

Smodels Returns:

$$\begin{array}{l} \{ p(2), d_2(2), d_1(1), a \} \\ \{ p(2), p(1), d_2(2), d_1(1), a \} \end{array}$$

# Implementation Details

Improved Mary's Implementation:

- any number of bound variables can be included in the sets.
- max and min aggregates
- sum, max and min of  $i^{th}$  bound variable.

## Example

Given a record of scores obtained by each student in a course as:  $\text{score}(s1,56,c1)$ ,  $\text{score}(s2,98,c1)$ , etc., We can find the average score of the class as:

$$\begin{aligned} \text{average}(A, C) : & - \text{course}(C), \\ & N = \text{card}(\{S, M : \text{score}(S, M, C)\}), \\ & T = \text{sum}(\{M, S : \text{score}(S, M, C)\}), \\ & A = T/N. \end{aligned}$$

# A Motivating Example

$q(0..100).$

$\{X : p(X)\} \subseteq \{X : q(X)\}.$

$a :- N = \text{sum}(\{X : p(X)\}), N < 100.$

The third rule will be grounded to get:

$a :- 0 = \text{sum}(\{X : p(X)\}).$

$\vdots$

$a :- 99 = \text{sum}(\{X : p(X)\}).$

of which, only one rule is fired.

## Another Example

$q(0..100)$ .

$\{X : p(X)\} \subseteq \{X : q(X)\}$ .

$psum(N) :- N = sum(\{X : p(X)\})$ .

# Implementation on Surya

currently, I am working implementing sets and aggregates on System Surya using these ideas.