# A-POL a Partial Order Programming extension for Answer Set Programming

Enrique Corona

Universidad de las Américas-Puebla

TAG Meeting, Las Cruces, New Mexico. August 21, 2003.

# Presentation Plan

- Preeliminaries
- A-POL Description
- Implementation
- Examples and results

# Preeliminaries

- Partial Order Programming
- Form of partial-order clauses:

  $f(terms) \geq expression :- p_1(terms_1), \ldots, p_n(terms_n).$

  $f(terms) \leq expression :- p_1(terms_1), \ldots, p_n(terms_n).$

- Partial Order Clauses as normal programs

# A-POL Description

- What is A-POL?
- Main advantages:
  - Efficient
  - Domain size
- Main features:
  - Witness recovery
  - User defined partial orders
  - User defined libraries

# A-POL Description

Shortest distance between two nodes of a graph

```
node(a).  node(b).  node(c).  node(d).

edge(a,b,1).  edge(b,c,2).  edge(a,d,4).

start(a).  end(c).

short(X)<= 0 :- start(X).

short(X)<= add(short(Y),C) :- edge(Y,X,C).
```

# A-POL Description

A-POL accepts disjunctive clauses of the form:

$$p_1(t_1) \vee \ldots \vee p_n(t_n) : -$$
$$b_1(t_{n+1}), \ldots, b_m(t_{n+m}),$$
$$f_1(t_{n+m+1}) = r_1, \ldots, f_s(t_{n+m+s}) = r_s.$$

and partial-order clauses of the form:

$$f(terms) \geq expression : -$$
$$p_1(terms_1), \ldots, p_n(terms_n). \ / \ info(iterms).$$
$$f(terms) \leq expression : -$$
$$p_1(terms_1), \ldots, p_n(terms_n). \ / \ info(iterms).$$

## Implementation

Main Computation Cycle:

```
A-POL(P) {
    order = getLevel(P)
    models = ∅
    for each cl ∈ order
        if (cl are disjunctive clauses)
            models = ASP(models,cl)
        else
            models = POP(models,cl)
    return models
}
```

# Implementation

- Evaluation path for a stratified program:

  $level(cl_1) = 1$

  $level(cl_2) = 2$

  $level(cl_3) = 3$

  ....

  $level(cl_n) = n$

- `ASP` - we use DLV[a]
- `POP` - general dynamic programming algorithm.

[a] http://www.dbai.tuwien.ac.at/proj/dlv/

# Examples and results

Shortest Path

```
declare < short(node).

node(a).  node(b).  node(c).  node(d).

edge(a,b,1).  edge(b,c,2).  edge(a,d,4).

start(a).  end(c).

short(X)<= 0 :- start(X).

short(X)<= add(short(Y),C) :- edge(Y,X,C). / way(Y,X).

select way where short(X) <- end(X).
```

# Examples and results

## DLV Shortest Path

```
way(X,Y,C) v other(X,Y,C) :- edge(X,Y,C).
:- edge(X,A,C), start(A), way(X, A,C).
:- edge(D,X,C), end(D), way(D,X,C).
:- edge(X,Y,C), edge(X,Y1,C1), way(X,Y,C),
    way(X,Y1,C1), Y != Y1.
:- edge(X,Y,C), edge(X1,Y,C1), way(X,Y,C),
    way(X1,Y,C1), X != X1.
r(X) :- start(X).
r(X) :- r(Y), edge(Y,X,C), way(Y,X,C).
:- end(D), not r(D).
:  way(X,Y,C). [C:1]
```

# Examples and results

Experimental results for shortest path implementations

| # nodes | DLV | A-POL |
|---|---|---|
| **10** | 0.16s | 0.65s |
| **20** | 4.77s | 0.84s |
| **40** | 4m 21.45s | 1.15s |
| **60** | 22m 57.02s | 2.21s |
| **80** | - | 2.85s |

# Examples and results

## 0-1 Knapsack

```
maxint = 20.

declare > ks(object,tweight).

declare > profit(object).

select take1 where ks(4,10).

object(0..4).  tweight(0..10).

weight(1,2).  weight(2,3).  weight(3,5).  weight(4,8).

profit(0)>= 0.  profit(1)>= 1.  profit(2)>= 2.

profit(3)>= 3.  profit(4)>= 5.

ks(N,K)>= ks(sub(N,1),K) :- le(1,N). / take1(N,K).

ks(N,K)>= add(profit(N),ks(sub(N,1),sub(K,X))) :-
          weight(N,X), le(X,K), le(1,N). / take1(N,K).

take(X1,Y) :- take1(X1,Y), not take(X2,Y), X2 > X1,
              object(X2), tweight(Y).
```

# Examples and results

Other problems coded in A-POL

- Optimal parentization for a matrix-chain multiplication
- Data-flow analysis equations
- ACM International Collegiate Programming Contest problems (fastfood, matrix multiplication)
- A simple capital budgeting problem[a]
- Cartoon Co. example from the DLV manual (introducing aggregate functions)

[a] http://mat.gsia.cmu.edu/classes/dynamic/node2.html

# Conclusions and future work

Conclusions

- Succefully implementation of A-POL
- Experimental results support our extension for ASP
- We present typical optimization problems coded in A-POL.

Future work

- Only a subset of DLV language is supported
- Explore the uses of partial-order clauses in other DLV front ends (like DLV-K).

# Thank you!

http://www.udlap.mx/ is108990/apol