Justification and Debugging of Answer Set Programs in ASP-PROLOG

Omar Elkhatib, Enrico Pontelli, Tran Cao Son

Knowledge representation, Logic, and Advanced Programming Laboratory
Department of Computer Science
New Mexico State University

Answer Set Programming (ASP)

- ASP: Logic Programming under answer set semantics
  - New Logic Programming Paradigm
  - Semantics of a Program = collection of answer sets (sets of atoms)
  - Rules $\text{Head} \leftarrow A_1, \ldots, A_n, \text{not } B_1, \ldots, \text{not } B_m$

    as constraints on admissible answer sets
  - Answer Sets of a Program P correspond to the solution of the problem
- Good Implementations (e.g., Smodels, DLV)
- However, No Debugging systems exists.

Debugging of ASP

- Very hard, because of its highly declarative nature.
- Most of the computational details are hidden from the programmer.
- Hard to understand the reasons of the solver’s outcomes.
- Tracing is one way of Debugging ASP:
  - Large search trees
  - Intermixed proofs of different atoms

3

Justification of ASP

- **Justification** is a new approach:
  - Creates proof graphs for each true atom
  - Creates counter-examples for false atoms
- Originally developed for well-founded semantics in XSB.
- In ASP, it provides a proof of why an atom is or is not in an answer set.
- We develop justification for ASP and integrate it into the ASP-PROLOG System.

4

ASP-PROLOG System

- It provide a tight and semantically well-defined integration of Prolog and Answer Set Programming (ASP).
- The combined system enhances the expressive power of ASP:
  - Dynamic ASP modules (add/remove rules)
  - Reasoning about ASP modules from Prolog
  - Reasoning about collections of answer sets from Prolog
- The system is developed using the module and class capabilities of CIAO Prolog.

System Download: www.cs.nmsu.edu/~okhatib/asp_prolog.html
Under Linux.

5

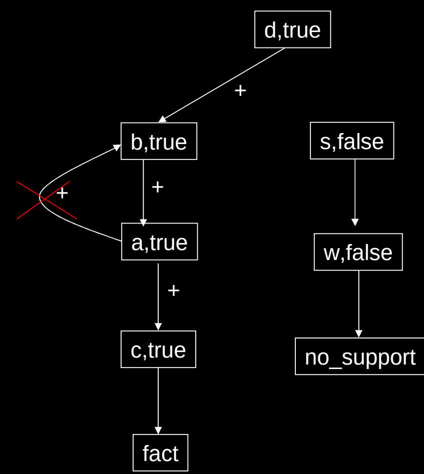
Justification of ASP Programs

- For ASP P and model M:
  - True literal L means:
    - L in M if L is an atom (L=a).
    - L not in M If L is a negated atom (L=not a).
  - False literal L means:
    - L not in M if L is an atom (L=a).
    - L in M if L is a negated atom (L=not a).
  - A rule r is **active** if all literals in body of r are true wrt M.
  - Locally consistent explanation (LCE):
    - $A \in M$: $\zeta(A,M)$ = set of the bodies of the *active* rules that have A as head (i.e., reasons for A's truth)
    - $A \notin M$: $\zeta(A,M)$ = a collection of literals such that we have exactly one false literal per rule for all rules which has head A (i.e., reasons for A's falsity)

6

Justification Example

a :- b.
b :- a.
a :- c.
c.
d :- s.
d :- b.
s :- a, w.



M = {a,b,c,d}
ζ(a,M)={{b},{c}}.
ζ(d,M)={{b}}.
ζ(s,M)={{w}}.

Positive cycles problem.

7

Justification of ASP

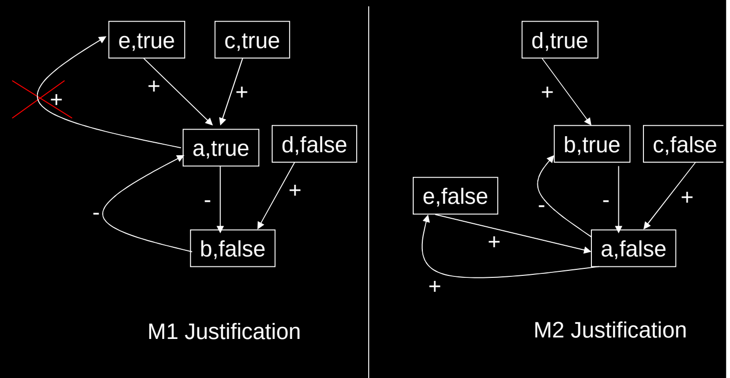
Justification of ASP P is a graph J=(V,E).

- If A in M:
  - If A is a fact then (A,fact) in E.
  - If there is rule r : all literals in body of r are not in a positive cycle with A, then (A, B) in E, ∀B in body of r.
  - No other outgoing edges from A are possible.
- If A not in M:
  - If no rule defined for A then (A,no_support)∈E
  - For each rule r with head A, choose one false literal B in body of r, then (A,B) in E.
  - No other outgoing edges from A are possible.

8

Justification Example

a :- not b.
b :- not a.
a :- e.
e :- a.
c :- a.
d :- b.
M1={a,c,e}.
M2={b,d}.



Positive and negative cycles.

9

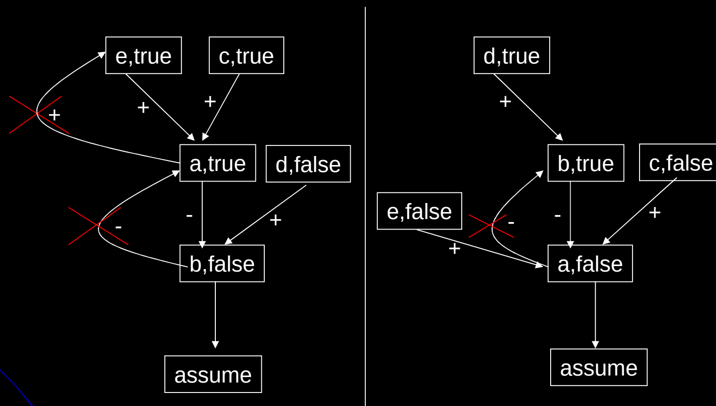
r-Justification

- Break all negative cycles.
- Define: Assumption Set $\mathcal{AS}$(M)= set of all atoms that satisfy the following conditions:
  - Atom A is false wrt M.
  - A appears in negation form in P.
  - A appears in a negative cycle in E
- R-justification:
  Add: If A$\in\mathcal{AS}$ then (A, assume)$\in$E.

10

r-Justification Example

a :- not b.
b :- not a.
a :- e.
e :- a.
c :- a.
d :- b.
M1={a,c,e}.
M2={b,d}.
$\mathcal{AS}$(M1)={b}.
$\mathcal{AS}$(M2)={a}.

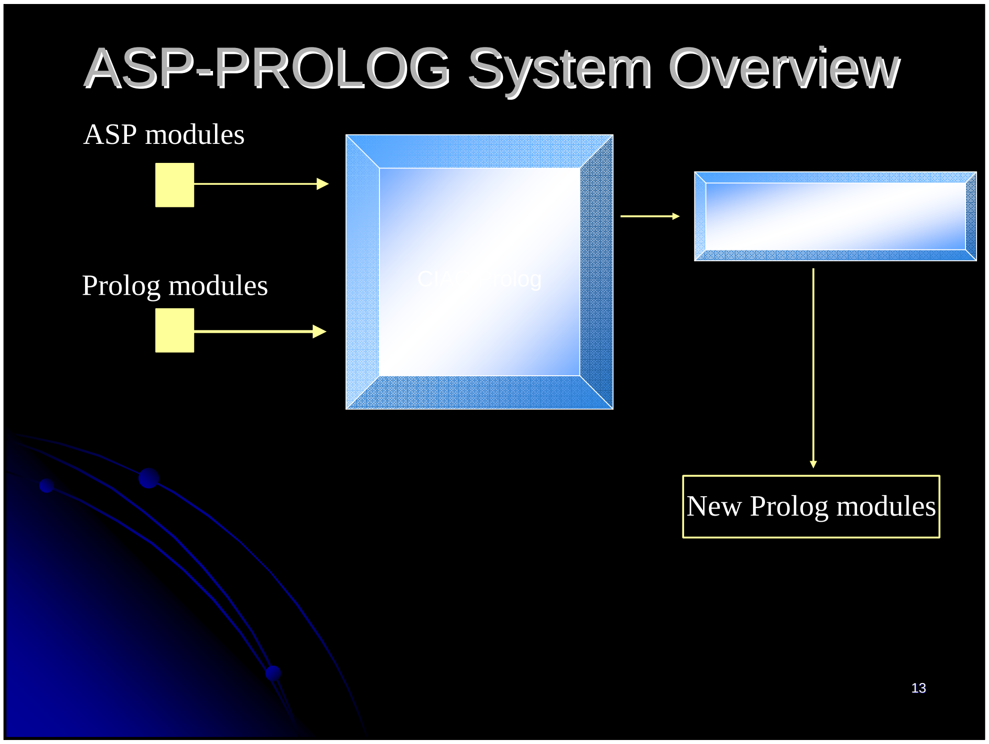


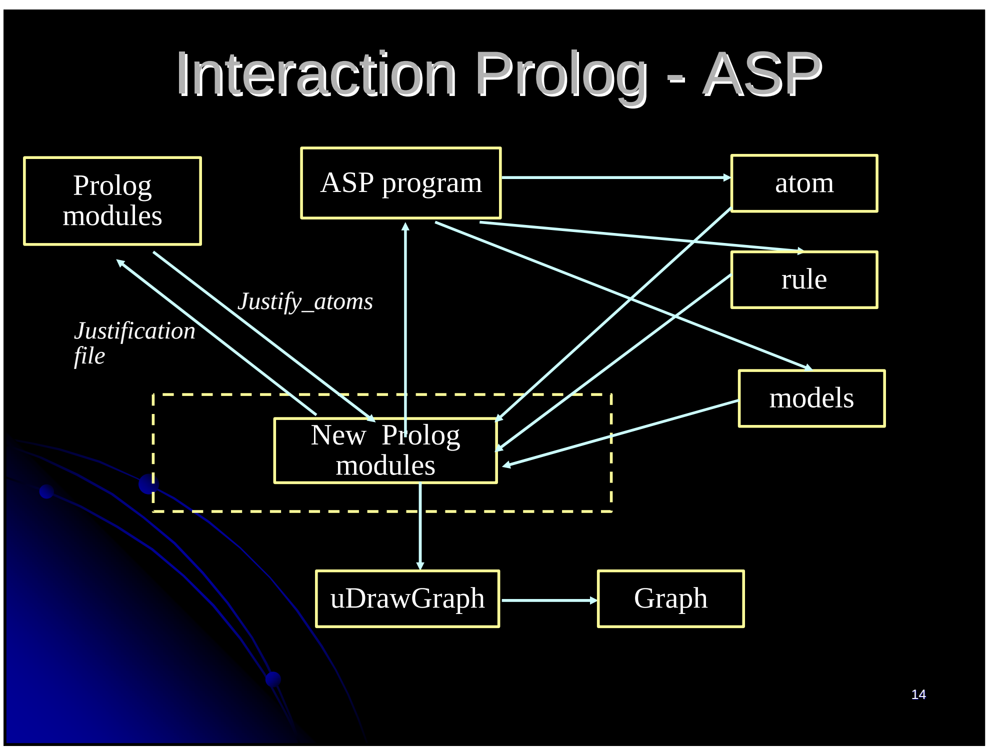
M1 Justification

M2 Justification

System Implementation

- Justification is integrated into ASP-PROLOG.
- Justification is written in CIAO-PROLOG.
- lparse/smodels is used to find answer set models.
- Predicate added for programmer:
  - Justify_atoms(model_name, atom_list).
  - Output: text format and graph format (uDrawGraph).
  - System shows the rules that cause the justification.
- System can handle all type of lparse/smodels rule: cardinality, weight and choice rules.

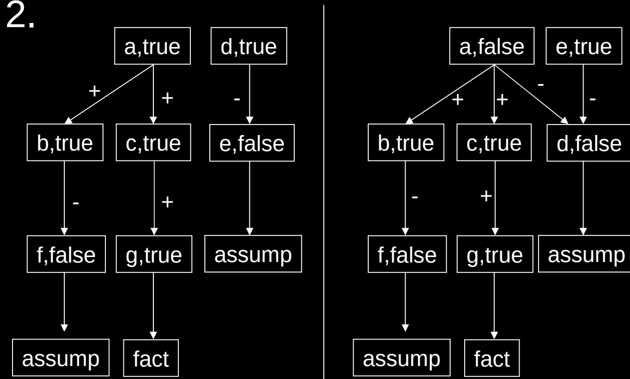

ASP-PROLOG System Overview
ASP modules
Prolog modules
New Prolog modules
13


Interaction Prolog - ASP
Prolog modules
ASP program
atom
rule
models
Justify_atoms
Justification file
New Prolog modules
uDrawGraph
Graph
14

Example

a :- 2 {b, c, not d} 2.
b :- not f.
f :- not b.
c :- g.
g.
d :- not e.
e :- not d.

M1={b,c,g,d,a}.
M2={b,c,g,e}.
M3={f,c,g,d}.
M4={f,c,g,e,a}.



M1 justification graph

M2 justification graph

15

Conclusion & Future Work

- Justification is one type of debugging. It is used in this paper to justify ASP models.
- Partial justification of answer sets is under investigation. Allow users to justify atoms in the middle of computation.
- Work is in progress to present the non-ground rules defining the atom.

16