Processing of Declarative Knowledge
–Normal Logic Programs–

Francesco Ricca

Computational Intelligence Curriculum
Institute of Information Systems

ASP Road map

ASP:

Datalog $\leftarrow$ done!

+ Default negation
+ Disjunction
+ Integrity Constraints
+ Weak Constraints
+ Aggregate atoms
+ ... and more

Datalog (followup)

Datalog: A logic language for querying databases

- overcomes some limits of Relational Algebra and SQL

  $\rightarrow$ Recursive definitions

- can be used for

  $\rightarrow$ Deductive database applications, query answering

- we have discussed some limits

  $\rightarrow$ e.g., limited usage of negation, no aggregation as in SQL, ...

Default Negation

Often, it is desirable to express negation in the following sense:

"If we do not have evidence that X holds, conclude Y."

This is expressed by default negation: the operator **not**.

Example (Cross railroad)

An agent could act according to the following rule:

```
% If the grass is not wet in the early morning,
% then conclude it did not rain in the night.

did_not_rain :- not wet_grass.
```

About Negation

Semantics:

- no negation $\rightarrow$ natural candidate: the minimal model
- with negation "unexpected" things may happen

About Models:

- consider

  a :– not b.

  b :– not a.

  $\rightarrow$ several minimal models $\{a\}$ and $\{b\}$
- also no minimal models
- but this may be... an advantage actually!

About Negation

Semantics:

- no negation $\rightarrow$ natural candidate: the minimal model
- with negation "unexpected" things may happen

About Models:

- consider

  a :– not b.

  b :– not a.

  $\rightarrow$ several minimal models $\{a\}$ and $\{b\}$
- also no minimal models
- but this may be... an advantage actually!

About Negation

Semantics:

- no negation $\rightarrow$ natural candidate: the minimal model
- with negation "unexpected" things may happen

About Models:

- consider

  a :– not b.

  b :– not a.

  $\rightarrow$ several minimal models $\{a\}$ and $\{b\}$
- also no minimal models
- but this may be... an advantage actually!

About Negation

Semantics:

- no negation $\rightarrow$ natural candidate: the minimal model
- with negation "unexpected" things may happen

About Models:

- consider

  a :– not b.

  b :– not a.

  $\rightarrow$ several minimal models $\{a\}$ and $\{b\}$
- also no minimal models
- but this may be... an advantage actually!

More than one model...

Observation:

- Several models represent several possible scenarios
- Several models are sets... several answer sets

Idea:

1. Represent a computational problem by a Logic program
2. Answer sets correspond to problem solutions
3. Use an ASP solver to find these solutions

More than one model...

Observation:

- Several models represent several possible scenarios
- Several models are sets... several answer sets

Idea:

1. Represent a computational problem by a Logic program
2. Answer sets correspond to problem solutions
3. Use an ASP solver to find these solutions

More than one model...

Observation:

- Several models represent several possible scenarios
- Several models are sets... several answer sets

Idea:

1. Represent a computational problem by a Logic program
2. Answer sets correspond to problem solutions
3. Use an ASP solver to find these solutions

Exercise 3SAT

Given a propositional formula Φ in 3 CNF, compute (if exists) an assignment to variables that satisfies Φ.

Can you encode it in Datalog?

Can you encode it in Datalog with a uniform fixed IDB?

Can you encode it with a ground Datalog program?

What if we consider unstratified programs...

...we will came back later to this!

Exercise 3SAT

Given a propositional formula Φ in 3 CNF, compute (if exists) an assignment to variables that satisfies Φ.

Can you encode it in Datalog?

Can you encode it in Datalog with a uniform fixed IDB?

Can you encode it with a ground Datalog program?

What if we consider unstratified programs...

...we will came back later to this!

Exercise 3SAT

Given a propositional formula Φ in 3 CNF, compute (if exists) an assignment to variables that satisfies Φ.

Can you encode it in Datalog?

Can you encode it in Datalog with a uniform fixed IDB?

Can you encode it with a ground Datalog program?

What if we consider unstratified programs...

...we will came back later to this!

Exercise 3SAT

Given a propositional formula Φ in 3 CNF, compute (if exists) an assignment to variables that satisfies Φ.

Can you encode it in Datalog?

Can you encode it in Datalog with a uniform fixed IDB?

Can you encode it with a ground Datalog program?

What if we consider unstratified programs...

...we will came back later to this!

Exercise 3SAT

Given a propositional formula Φ in 3 CNF, compute (if exists) an assignment to variables that satisfies Φ.

Can you encode it in Datalog?

Can you encode it in Datalog with a uniform fixed IDB?

Can you encode it with a ground Datalog program?

What if we consider unstratified programs...

...we will came back later to this!

Exercise 3SAT

Given a propositional formula Φ in 3 CNF, compute (if exists) an assignment to variables that satisfies Φ.

Can you encode it in Datalog?

Can you encode it in Datalog with a uniform fixed IDB?

Can you encode it with a ground Datalog program?

What if we consider unstratified programs...

...we will came back later to this!

Normal Logic Programs (propositional case)

Rule: (r) $\underbrace{a}_{head}$:- $\underbrace{b_1, \ldots, b_k, \text{not } b_{k+1}, \ldots, \text{not } b_m.}_{body}$

Intuitively:

"a is true if $b_1, \ldots, b_n$ are true and $b_{k+1}, \ldots, b_m$. are false"

Atoms and Literals: a_i , b_i, not b_i
Head of r: $H(r) = a$
Body of r: $B(r) = B^+(r) \cup B^-(r)$
Positive Body: $B^+(r) = \{b_1, \ldots, b_k\}$
Negative Body: $B^-(r) = \{\text{not } b_{k+1}, \ldots, \text{not } b_m.\}$

Fact: A rule with empty body
Variables: no variables, consider ground programs for now...
Negation: unrestricted

Normal Logic Programs (propositional case)

Rule: (r) $\underbrace{a}_{head} \text{ :- } \underbrace{b_1, \ldots, b_k, \text{not } b_{k+1}, \ldots, \text{not } b_m.}_{body}$

Intuitively:

"a is true if $b_1, \ldots, b_n$ are true and $b_{k+1}, \ldots, b_m$. are false"

Atoms and Literals: a_i , b_i, not b_i
Head of r: $H(r) = a$
Body of r: $B(r) = B^+(r) \cup B^-(r)$
Positive Body: $B^+(r) = \{b_1, \ldots, b_k\}$
Negative Body: $B^-(r) = \{\text{not } b_{k+1}, \ldots, \text{not } b_m.\}$

Fact: A rule with empty body
Variables: no variables, consider ground programs for now...
Negation: unrestricted

Formal Semantics: Roadmap

1) Positive Programs

2) Negative Programs

$\rightarrow$ **via Gelfong & Lifschitz Reduct**

Semantics (Ground) Positive Programs

Interpretation:
A set I of ground atoms, and atom a is true w.r.t. I if $a \in I$*, it is false otherwise.*
A negative literal not *a is true w.r.t. I if* $a \notin I$*, false otherwise.*

Satisfaction:
Rule r is satisfied w.r.t. I if $H(r) \in I$ *whenever all literals* $\ell \in B(r)$ *are true w.r.t. I*

Model:
Interpretation I is a model for program P if all rules in P are satisfied by I

Semantics (Ground) Positive Programs

Immediate consequence operator:
$T_P(I) = \{a \in H(r) : \forall b \in B(r), b \in I\}$

Least Model (or Answer Set):
Least fixpoint $LM(P)$ of T_P operator
$(T_P(\emptyset) \subseteq T_P(T_P(\emptyset)) \subseteq \cdots \subseteq LM(P) = T_P(LM(P)))$

Theorem:
A positive program P has a unique least model $M = LM(P)$ which is minimal under subset inclusion, actually $M = \cap_{I \in ModelsOf(P)} I$

Semantics for Programs with Negation

Consider *general* programs with negation

Reduct: The *Gelfond-Lifschitz reduct* of a program P w.r.t. an interpretation I is the positive program P^I obtained from P by:

- deleting all rules with a negative literal false in I;
- deleting the negative literals from the bodies of the remaining rules.

Answer Set: An *answer set* or *stable model* of a general program P^I is an interpretation I such that I is an answer set of P^I, i.e., $I = LM(P^I)$.

Example 1

Example (Reduct)

Program:

a :– d, not b.
b :– not d.

d.

Consider: $I = \{a, d\}$

Reduct:

a :– d.
d.

$\rightarrow$ *I is an answer set of* P^I *and therefore it is an answer set of* P.

Example 2

Example

Program:

a :– not b.

Answer Set: $\{a\}$

Example 3

Example

Program:

a :– not b.

b :– not a.

Answer Sets: $\{a\}, \{b\}$

$\rightarrow$ *Prolog would loop!*

Example 4

Example

Program:

a :– not b.
b :– not a.
c :– b.
c :– a.

Answer Sets: $\{a, c\}, \{b, c\}$

Example 5

Example

Program:

a :– not a.

Answer Set: no answer set!

Example 6

Example

Program:

a :– not b.
b :– not a.
f :– b, not f

Answer Set: $\{a\}$

Supported Models and Answer Sets (1)

Supported Model:

A model M is supported if for each $a \in M$ there exist rule $r \in P$ such that $H(r) = a$ and $\forall b \in B(r)$, b is true w.r.t. M

Intuition: Something is true if there is a rule "supporting" its truth.

Theorem:

Answer sets are supported models

Supported Models and Answer Sets (1)

Supported Model:

A model M is supported if for each $a \in M$ there exist rule $r \in P$ such that $H(r) = a$ and $\forall b \in B(r)$, b is true w.r.t. M

Intuition: Something is true if there is a rule “supporting” its truth.

Theorem:

Answer sets are supported models

Supported Models and Answer Sets (2)

Example (Inverse does not hold.)

Program:

$a \text{ :- } a.$

Models: $\{\}, \{a\}$ ← both are supported

Answer Set: $\{\}$

→ **Circular support is not allowed!**

→ **Empty answer set is fine!**

Supported Models and Answer Sets (2)

Example (Inverse does not hold.)

Program:

$a \text{ :- } a.$

Models: $\{\}, \{a\}$ ← both are supported

Answer Set: $\{\}$

→ **Circular support is not allowed!**

→ **Empty answer set is fine!**

Supported Models and Answer Sets (2)

Example (Inverse does not hold.)

Program:

$a \text{ :- } a.$

Models: $\{\}, \{a\}$ ← both are supported

Answer Set: $\{\}$

→ **Circular support is not allowed!**

→ **Empty answer set is fine!**

Unfounded Sets and Answer Sets (intuition)

Unfounded Set:

A set of ground atoms X is an unfounded set if, for each rule r s.t. $H(r) \in X$, one of the following conditions hold

1. the body of r is false, or
2. some literal in the positive body belongs to X

Example: $a :- a.$ and $X = \{a\}.$ is unfounded!

Theorem:

Answer sets are unfounded-free interpretations, i.e., no subset is unfounded.

Unfounded Sets and Answer Sets (intuition)

Unfounded Set:

A set of ground atoms X is an unfounded set if, for each rule r s.t. $H(r) \in X$, one of the following conditions hold

1. the body of r is false, or
2. some literal in the positive body belongs to X

Example: $a :\!\!- a.$ and $X = \{a\}.$ is unfounded!

Theorem:

Answer sets are unfounded-free interpretations, i.e., no subset is unfounded.

Exercise 3SAT

Given a propositional formula Φ in 3 CNF, compute an assignment to variables that satisfies Φ if it exists.

Write a logic program $P(\Phi)$ such that answer sets of $P(\Phi)$ correspond to satisfying assignments of Φ

Exercise 3SAT

Given a propositional formula Φ in 3 CNF, compute an assignment to variables that satisfies Φ if it exists.

Write a logic program $P(\Phi)$ such that answer sets of $P(\Phi)$ correspond to satisfying assignments of Φ