Systems and Solving Techniques for Knowledge Representation

– Normal Logic Programs –

Marco Maratea
University of Genoa, Italy

066 011 Double degree programme Computational Logic (Erasmus-Mundus)
066 931 Computational Intelligence
066 937 Software Engineering & Internet Computing
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 seen some limitations

  $\rightarrow$ 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

  a :– not a.

About Negation

Semantics:

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

About Models:

- consider

  $a \coloneq \text{not } b.$

  $b \coloneq \text{not } a.$

  $\rightarrow$ several minimal models $\{a\}$ and $\{b\}$
- also no minimal models

  $a \coloneq \text{not } a.$

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

  a :– not a.

About Negation

Semantics:

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

About Models:

- consider

  $a \text{ :- not } b.$

  $b \text{ :- not } a.$

  $\rightarrow$ several minimal models $\{a\}$ and $\{b\}$
- also no minimal models

  $a \text{ :- not } a.$

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

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, $\text{not } b_i$
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...
Safety: variables must occur in the positive body
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...
Safety: variables must occur in the positive body
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...
Safety: variables must occur in the positive body
Negation: unrestricted

Formal Semantics: Roadmap

1) Positive Programs

2) Programs with negation

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

Semantics for (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 $\mathrm{not}\ a$ 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 for (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 $M = \bigcap_{I \in ModelsOf(P)} I$

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 w.r.t. 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 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 P:

$a :- d, \text{not}\ \ b.$
$b :- \text{not}\ \ d.$

$d.$

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

Reduct P^I:

$a :- d.$
$d.$

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

Example 2

Example

Program:

$a \text{:-} \text{not } b.$

Answer Set: $\{a\}$

Example 3

Example

(Non-stratified) Program:

$a \mathrel{:-} \text{not}\ b.$

$b \mathrel{:-} \text{not}\ a.$

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

Example 4

Example

Program:

$$a :- \text{not } b.$$
$$b :- \text{not } a.$$
$$c :- b.$$
$$c :- a.$$

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

Example 5

Example

Program:

$a :- \text{not } a.$

Answer Set: no answer set!

Example 6

Example

Program:

$a :- \text{not } b.$
$b :- \text{not } a.$
$f :- b, \text{not } f$

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 (Converse does not hold.)

Program:

$a \coloneq a.$

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

Answer Set: $\{\}$

$\rightarrow$ **Circular support is not allowed!**

$\rightarrow$ **Empty answer set is fine!**

Supported Models and Answer Sets (2)

Example (Converse does not hold.)

Program:

$a \mathrel{:\!-} a.$

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

Answer Set: $\{\}$

$\rightarrow$ **Circular support is not allowed!**

$\rightarrow$ **Empty answer set is fine!**

Supported Models and Answer Sets (2)

Example (Converse does not hold.)

Program:

$a :\!-\, a.$

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

Answer Set: $\{\}$

$\rightarrow$ **Circular support is not allowed!**

$\rightarrow$ **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 w.r.t. X, or
2. some literal in the positive body belongs to X.

Example: In program a :- a., $X = \{a\}$ 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 w.r.t. X, or
2. some literal in the positive body belongs to X.

Example: In program $a \,\text{:-}\, a.$, $X = \{a\}$ is unfounded!

Theorem:

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

Thanks to Francesco Ricca for a preliminary version of these slides