Answer Set Programming

Wolfgang Faber

University of Calabria, Italy
soon: University of Huddersfield, UK
wf@wfaber.com

RW2013, Mannheim, Germany

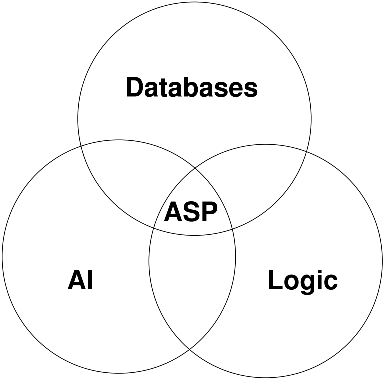
Outline

Outline

Part I

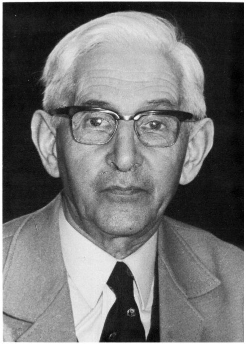
From Datalog to Answer Set Programming

Setting



Early Roots: Constructive Logic

Intuitionistic or Constructive Logic



Luitzen Egbertus Jan Brouwer   Arend Heyting

Early Roots: Game Theory

Stability conditions in mathematical games and economy

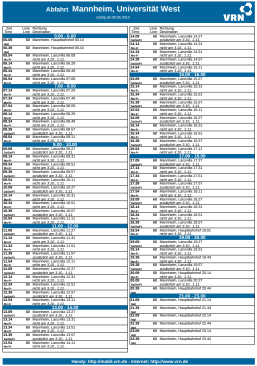
Oskar Morgenstern John Von Neumann

Difference to Classical Logic

Main Differences to Classical Logic:

- Closed World Assumption
    - Implicit Necessity
- Unique Name Assumption
    - Unique Identifiers

Closed World Assumption



Abfahrt Mannheim, Universität West

Gültig ab 09.06.2013

VRN

Zeit / Time	Linie / Line	Richtung / Destination
0.00 - 6.00		
00.09 tägl	60	Mannheim, Hauptbahnhof 00.14
00.39 tägl	60	Mannheim, Hauptbahnhof 00.44
05.54 Mo-Fr	60	Mannheim, Lanzvilla 06.09
06.14 Mo-Fr	60	Mannheim, Lanzvilla 06.29 *nicht am 3.10., 1.11.*
06.34 Mo-Fr	60	Mannheim, Lanzvilla 06.49 *nicht am 3.10., 1.11.*
06.54 Mo-Fr	60	Mannheim, Lanzvilla 07.09 *nicht am 3.10., 1.11.*
7.00 - 8.00		
07.14 Mo-Fr	60	Mannheim, Lanzvilla 07.29 *nicht am 3.10., 1.11.*
07.34 Mo-Fr	60	Mannheim, Lanzvilla 07.49 *nicht am 3.10., 1.11.*
07.54 Mo-Fr	60	Mannheim, Lanzvilla 08.09 *nicht am 3.10., 1.11.*
08.14 Mo-Fr	60	Mannheim, Lanzvilla 08.29 *nicht am 3.10., 1.11.*
08.34 Mo-Fr	60	Mannheim, Lanzvilla 08.49 *nicht am 3.10., 1.11.*
08.39 Sa/So/Ft	60	Mannheim, Lanzvilla 08.57 *zusätzlich am 3.10., 1.11.*
08.54 Mo-Fr	60	Mannheim, Lanzvilla 09.11 *nicht am 3.10., 1.11.*
9.00 - 10.00		
09.09 Sa/So/Ft	60	Mannheim, Lanzvilla 09.27 *zusätzlich am 3.10., 1.11.*
09.14 Mo-Fr	60	Mannheim, Lanzvilla 09.31 *nicht am 3.10., 1.11.*
09.34 Mo-Fr	60	Mannheim, Lanzvilla 09.51 *nicht am 3.10., 1.11.*
09.39 Sa/So/Ft	60	Mannheim, Lanzvilla 09.57 *zusätzlich am 3.10., 1.11.*
09.54 Mo-Fr	60	Mannheim, Lanzvilla 10.11 *nicht am 3.10., 1.11.*
10.09 Sa/So/Ft	60	Mannheim, Lanzvilla 10.27 *zusätzlich am 3.10., 1.11.*
10.14 Mo-Fr	60	Mannheim, Lanzvilla 10.31 *nicht am 3.10., 1.11.*
10.34 Mo-Fr	60	Mannheim, Lanzvilla 10.51 *nicht am 3.10., 1.11.*
10.39 Sa/So/Ft	60	Mannheim, Lanzvilla 10.57 *zusätzlich am 3.10., 1.11.*
10.54 Mo-Fr	60	Mannheim, Lanzvilla 11.11 *nicht am 3.10., 1.11.*
11.00 - 12.00		
11.09 Sa/So/Ft	60	Mannheim, Lanzvilla 11.27 *zusätzlich am 3.10., 1.11.*
11.14 Mo-Fr	60	Mannheim, Lanzvilla 11.31 *nicht am 3.10., 1.11.*
11.34 Mo-Fr	60	Mannheim, Lanzvilla 11.51 *nicht am 3.10., 1.11.*
11.39 Sa/So/Ft	60	Mannheim, Lanzvilla 11.57 *zusätzlich am 3.10., 1.11.*
11.54 Mo-Fr	60	Mannheim, Lanzvilla 12.11 *nicht am 3.10., 1.11.*
12.09 Sa/So/Ft	60	Mannheim, Lanzvilla 12.27 *zusätzlich am 3.10., 1.11.*
12.14 Mo-Fr	60	Mannheim, Lanzvilla 12.31 *nicht am 3.10., 1.11.*
12.34 Mo-Fr	60	Mannheim, Lanzvilla 12.51 *nicht am 3.10., 1.11.*
12.39 Sa/So/Ft	60	Mannheim, Lanzvilla 12.57 *zusätzlich am 3.10., 1.11.*
12.54 Mo-Fr	60	Mannheim, Lanzvilla 13.11 *nicht am 3.10., 1.11.*
13.00 - 14.00		
13.09 Sa/So/Ft	60	Mannheim, Lanzvilla 13.27 *zusätzlich am 3.10., 1.11.*
13.14 Mo-Fr	60	Mannheim, Lanzvilla 13.31 *nicht am 3.10., 1.11.*
13.34 Mo-Fr	60	Mannheim, Lanzvilla 13.51 *nicht am 3.10., 1.11.*
13.39 Sa/So/Ft	60	Mannheim, Lanzvilla 13.57 *zusätzlich am 3.10., 1.11.*
13.54 Mo-Fr	60	Mannheim, Lanzvilla 14.11 *nicht am 3.10., 1.11.*
14.09 Sa/So/Ft	60	Mannheim, Lanzvilla 14.27 *zusätzlich am 3.10., 1.11.*
14.14 Mo-Fr	60	Mannheim, Lanzvilla 14.31 *nicht am 3.10., 1.11.*
14.34 Mo-Fr	60	Mannheim, Lanzvilla 14.51 *nicht am 3.10., 1.11.*
14.39 Sa/So/Ft	60	Mannheim, Lanzvilla 14.57 *zusätzlich am 3.10., 1.11.*
14.54 Mo-Fr	60	Mannheim, Lanzvilla 15.11 *nicht am 3.10., 1.11.*
15.00 - 16.00		
15.09 Sa/So/Ft	60	Mannheim, Lanzvilla 15.27 *zusätzlich am 3.10., 1.11.*
15.14 Mo-Fr	60	Mannheim, Lanzvilla 15.31 *nicht am 3.10., 1.11.*
15.34 Mo-Fr	60	Mannheim, Lanzvilla 15.51 *nicht am 3.10., 1.11.*
15.39 Sa/So/Ft	60	Mannheim, Lanzvilla 15.57 *zusätzlich am 3.10., 1.11.*
15.54 Mo-Fr	60	Mannheim, Lanzvilla 16.11 *nicht am 3.10., 1.11.*
16.09 Sa/So/Ft	60	Mannheim, Lanzvilla 16.27 *zusätzlich am 3.10., 1.11.*
16.14 Mo-Fr	60	Mannheim, Lanzvilla 16.31 *nicht am 3.10., 1.11.*
16.34 Mo-Fr	60	Mannheim, Lanzvilla 16.51 *nicht am 3.10., 1.11.*
16.39 Sa/So/Ft	60	Mannheim, Lanzvilla 16.57 *zusätzlich am 3.10., 1.11.*
16.54 Mo-Fr	60	Mannheim, Lanzvilla 17.11 *nicht am 3.10., 1.11.*
17.00 - 18.00		
17.09 Sa/So/Ft	60	Mannheim, Lanzvilla 17.27 *zusätzlich am 3.10., 1.11.*
17.14 Mo-Fr	60	Mannheim, Lanzvilla 17.31 *nicht am 3.10., 1.11.*
17.34 Mo-Fr	60	Mannheim, Lanzvilla 17.51 *nicht am 3.10., 1.11.*
17.39 Sa/So/Ft	60	Mannheim, Lanzvilla 17.57 *zusätzlich am 3.10., 1.11.*
17.54 Mo-Fr	60	Mannheim, Lanzvilla 18.11 *nicht am 3.10., 1.11.*
18.09 Sa/So/Ft	60	Mannheim, Lanzvilla 18.27 *zusätzlich am 3.10., 1.11.*
18.14 Mo-Fr	60	Mannheim, Lanzvilla 18.31 *nicht am 3.10., 1.11.*
18.34 Mo-Fr	60	Mannheim, Lanzvilla 18.51 *nicht am 3.10., 1.11.*
18.39 Sa/So/Ft	60	Mannheim, Lanzvilla 18.57 *zusätzlich am 3.10., 1.11.*
18.54 Mo-Fr	60	Mannheim, Hauptbahnhof 19.01 *nicht am 3.10., 1.11.*
19.00 - 20.00		
19.09 Sa/So/Ft	60	Mannheim, Lanzvilla 19.27 *zusätzlich am 3.10., 1.11.*
19.14 Mo-Fr	60	Mannheim, Lanzvilla 19.31 *nicht am 3.10., 1.11.*
19.39 Mo-Fr	60	Mannheim, Hauptbahnhof 19.44 *nicht am 3.10., 1.11.*
19.39 Sa/So/Ft	60	Mannheim, Lanzvilla 19.57 *zusätzlich am 3.10., 1.11.*
20.09 Mo-Fr	60	Mannheim, Hauptbahnhof 20.14 *nicht am 3.10., 1.11.*
20.09 Sa/So/Ft	60	Mannheim, Lanzvilla 20.27 *zusätzlich am 3.10., 1.11.*
20.39 tägl	60	Mannheim, Hauptbahnhof 20.44
21.00 - 23.00		
21.09 tägl	60	Mannheim, Hauptbahnhof 21.14
21.39 tägl	60	Mannheim, Hauptbahnhof 21.44
22.09 tägl	60	Mannheim, Hauptbahnhof 22.14
22.39 tägl	60	Mannheim, Hauptbahnhof 22.44
23.09 tägl	60	Mannheim, Hauptbahnhof 23.14
23.39 tägl	60	Mannheim, Hauptbahnhof 23.44

Handy: http://mobil.vrn.de - Internet: http://www.vrn.de

Is there a bus scheduled at 9.34? At 9.40?

Closed World Assumption

		9.00 - 10.00
09.09 Sa/So/Ft	60	Mannheim, Lanzvilla 09.27 *zusätzlich am 3.10., 1.11.*
09.14 Mo-Fr	60	Mannheim, Lanzvilla 09.31 *nicht am 3.10., 1.11.*
09.34 Mo-Fr	60	Mannheim, Lanzvilla 09.51 *nicht am 3.10., 1.11.*
09.39 Sa/So/Ft	60	Mannheim, Lanzvilla 09.57 *zusätzlich am 3.10., 1.11.*
09.54 Mo-Fr	60	Mannheim, Lanzvilla 10.11 *nicht am 3.10., 1.11.*
10.09 Sa/So/Ft	60	Mannheim, Lanzvilla 10.27 *zusätzlich am 3.10., 1.11.*
10.14 Mo-Fr	60	Mannheim, Lanzvilla 10.31 *nicht am 3.10., 1.11.*
10.34 Mo-Fr	60	Mannheim, Lanzvilla 10.51 *nicht am 3.10., 1.11.*
10.39 Sa/So/Ft	60	Mannheim, Lanzvilla 10.57 *zusätzlich am 3.10., 1.11.*
10.54 Mo-Fr	60	Mannheim, Lanzvilla 11.11 *nicht am 3.10., 1.11.*
		11.00 - 12.00

Is there a bus scheduled at 9.34? At 9.40?

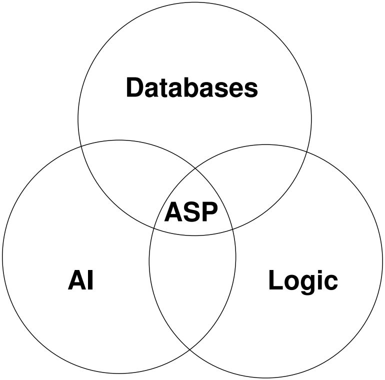
Closed World Assumption

9.00 - 10.00		
09.09 Sa/So/Ft	60	Mannheim, Lanzvilla 09.27 *zusätzlich am 3.10., 1.11.*
09.14 Mo-Fr	60	Mannheim, Lanzvilla 09.31 *nicht am 3.10., 1.11.*
09.34 Mo-Fr	60	Mannheim, Lanzvilla 09.51 *nicht am 3.10., 1.11.*
09.39 Sa/So/Ft	60	Mannheim, Lanzvilla 09.57 *zusätzlich am 3.10., 1.11.*
09.54 Mo-Fr	60	Mannheim, Lanzvilla 10.11 *nicht am 3.10., 1.11.*
10.09 Sa/So/Ft	60	Mannheim, Lanzvilla 10.27 *zusätzlich am 3.10., 1.11.*
10.14 Mo-Fr	60	Mannheim, Lanzvilla 10.31 *nicht am 3.10., 1.11.*
10.34 Mo-Fr	60	Mannheim, Lanzvilla 10.51 *nicht am 3.10., 1.11.*
10.39 Sa/So/Ft	60	Mannheim, Lanzvilla 10.57 *zusätzlich am 3.10., 1.11.*
10.54 Mo-Fr	60	Mannheim, Lanzvilla 11.11 *nicht am 3.10., 1.11.*
11.00 - 12.00		

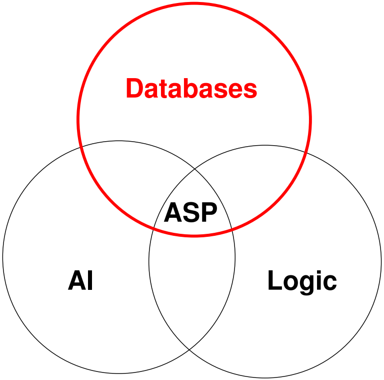
Is there a bus scheduled at 9.34? At 9.40?

Outline

Setting

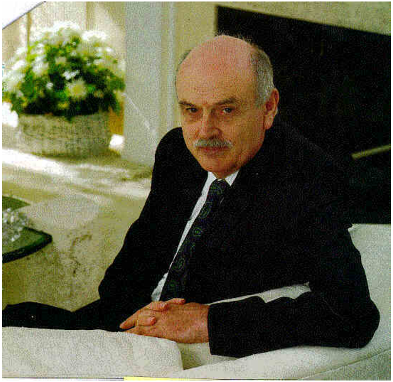


Setting



Relational Model

Relational Model – Codd 1970



Edgar Frank Codd

Relations

- Schema:
  - Domain (denumerable set)
  - Attributes (denumerable set)
  - Relations (subset of attributes)
- Instances:
  - Relation instances: Sets of tuples.
  - Each tuple is a function from the relation's attributes to domain elements.
  - Database instance: Collection of relation instances.

Relations: Example

$$A = \{X, Y\}, D = \{a, b, c, d\}$$
$$R = \{X, Y\}, S = \{Y\}$$

$$I(R) = \{t_1, t_2\}$$
$$t_1(X) = a, t_1(Y) = b, t_2(X) = c, t_2(Y) = d$$
$$I(S) = \{t_3\}, t_3(Y) = d$$

$$I(R) = \{\langle a, b\rangle, \langle c, d\rangle\}, I(S) = \{\langle d\rangle\}$$

Relations: Example

R	X	Y
	a	b
	c	d

S	Y
	d

Relational Algebra

Basic Operators:

- σ Selection
- π Projection
- $\times$ Cartesian Product
- $\cup$ Union
- $-$ Difference

Definable using Basic Operators:

- $\bowtie$ Join [$R \bowtie S = \sigma_F(R \times S)$]
- $\ltimes$ Semijoin [$R \ltimes S = \pi_{Schema(R)}(R \bowtie S)$]
- $\cap$ Intersection

Relational Algebra Example

$R \times S$	X_R	Y_R	Y_S
	a	b	d
	c	d	d

$\sigma_{2=3}(R \times S)$	X_R	Y_R	Y_S
	c	d	d

$\pi_{1,2}(\sigma_{2=3}(R \times S))$	X	Y
	c	d

Outline

Relations – Logical View

- Schema:
  - Domain – Constant symbols (denumerable set)
  - Relations – Predicate symbols (attributes are not explicitly named)
  - Attributes – implicit by predicate arity
- Instances:
  - Relation instances: Subset of ground instances for relation predicate.
  - Database instance: Subset of Herbrand Base.

Relations: Example

$$D = \{a, b, c, d\}$$
$$R/2, S/1$$
$$I(R) = \{R(a, b), R(c, d)\}, I(S) = \{S(d)\}$$
$$I = \{R(a, b), R(c, d), S(d)\}$$

Relational Calculus

- Based on First-Order Logic
- Atomic formulas $r(X_1, \ldots, X_n)$
- Comparison formulas $X = 2$ or $X = Y$ (pre-interpreted predicate)
- Composed formulas using $\neg$, $\wedge$, $\exists$
- $\rightarrow$, $\leftrightarrow$, $\vee$, $\forall$ added as "syntactic sugar"

Relational Calculus

- Relational Algebra expressions represent relation instances
- In Relational Calculus: $\{e_1, \ldots, e_n \mid \phi\}$
  - ϕ is a Relational Calculus formula
  - $e_1, \ldots, e_n$: terms containing exactly the free variables of ϕ
- Collect all substitutions for free variables such that ϕ is true in the interpretation formed by the database.
- The defined relation is obtained by applying all of these substitutions to $e_1, \ldots, e_n$.

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Relational Calculus Examples

$$\{X, Y, Z \mid R(X, Y) \wedge S(Z)\} = \{T(a, b, d), T(c, d, d)\} = R \times S$$

$$\{X, Y, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d, d)\} = \sigma_{2=3}(R \times S)$$

$$\{X, Y \mid R(X, Y) \wedge S(Y)\} = \{T(c, d)\} = \pi_{1,2}(\sigma_{2=3}(R \times S))$$

Algebra as Calculus

- σ_S r $\{X_1, \ldots, X_n \mid r(X_1, \ldots, X_n) \wedge S\}$
- π_i r $\{X_i \mid \exists X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n : r(X_1, \ldots, X_n)\}$
- r $\times$ s
  $\{X_1, \ldots, X_n, Y_1, \ldots, Y_m \mid r(X_1, \ldots, X_n) \wedge s(Y_1, \ldots, Y_m)\}$
- r $\cup$ s $\{X_1, \ldots, X_n \mid r(X_1, \ldots, X_n) \vee s(X_1, \ldots, X_n)\}$
- r $-$ s $\{X_1, \ldots, X_n \mid r(X_1, \ldots, X_n) \wedge \neg s(X_1, \ldots, X_n)\}$

Outline

Calculus: More then Algebra

Problematic expressions:

$$\{X \mid \neg R(a, X)\}$$
$$\{X, Y \mid R(a, X) \vee R(Y, b)\}$$
$$\{X \mid \forall Y : R(X, Y)\}$$

Calculus: More then Algebra

Using the domain of the database:

- $\{X \mid \neg R(a, X)\}$
  - all constants c of the domain such that (a, c) is no tuple in R
  - will be infinite if the domain is infinite
- $\{X, Y \mid R(a, X) \vee R(Y, b)\}$
  - if R contains some tuple (a, b), the result is (b, c) for all constants c in the domain
  - will be infinite if the domain is infinite
- $\{X \mid \forall Y : R(X, Y)\}$
  - this will be always empty if the domain is infinite, because relations are finite

Calculus: More then Algebra

Using the active domain of the database (only constants appearing in the database and the query):

- $\{X \mid \neg R(a, X)\}$
  - all constants c in the database such that (a, c) is no tuple in R
  - will change if some unrelated constant is added
- $\{X, Y \mid R(a, X) \vee R(Y, b)\}$
  - if R contains some tuple (a, b), the result is (b, c) for all constants c in the database
  - will change if some unrelated constant is added
- $\{X \mid \forall Y : R(X, Y)\}$
  - will unintuitively become empty if an unrelated constant is added

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Natural versus Active Domain Semantics

1. Natural Semantics: Interpretations from Database Domain
   - pro: Classical First-Order theory
   - contra: Produces infinite relations
   - contra: Quantification over infinite sets
2. Active Domain Semantics: Interpretations from Active Domain
   - pro: Always finite
   - contra: Frequently gives unintuitive results
   - contra: Active Domain not always available

Domain Independent Queries

Idea: Consider only those queries for which Natural and Active Domain Semantics coincide.

Definition

A query in the relational calculus is domain independent, if it yields the same answer using the natural (full) domain and the active domain.

Domain Independent Queries

Idea: Consider only those queries for which Natural and Active Domain Semantics coincide.

Definition

A query in the relational calculus is domain independent, if it yields the same answer using the natural (full) domain and the active domain.

Domain Independent Queries

Theorem

Any query of the Relational Algebra can be written as a domain independent query of Relational Calculus, and vice versa.

Domain Independent Queries

Theorem

Any query of the Relational Algebra can be written as a domain independent query of Relational Calculus, and vice versa.

Great, let's use only domain independent queries of Relational Calculus!

Domain Independent Queries

Theorem

Deciding whether a query of Relational Calculus is domain independent, is undecidable.

Safe Range Queries

Define a syntactically restricted fragment of Relational Calculus queries, which is guaranteed to be domain independent.

1. Transform formula into a normal form (SRNF).
2. Determine range restricted variables of the SRNF formula.
3. Check whether the range restricted variables are exactly the free variables.

SRNF

- Normalize variables: Rename variables, so that each quantifier binds a distinct variable and free and bound variables are different.
- Remove $\forall$: $\forall X : \phi \Rightarrow \neg\exists X : \neg\phi$
- Remove $\rightarrow$: $\phi \rightarrow \psi \Rightarrow \neg\phi \vee \psi$
- Remove $\neg\neg$: $\neg\neg\phi \Rightarrow \phi$
- Push $\neg$: $\neg(\phi \wedge \psi) \Rightarrow (\neg\phi \vee \neg\psi)$
- Push $\neg$: $\neg(\phi \vee \psi) \Rightarrow (\neg\phi \wedge \neg\psi)$

Apply these rules as until none is applicable.

Range Restricted Variables

Intuition: In formulas, recursively determine variables, for which the value is determined by the database instance.

- Equality needs caution
- Disjunction?
- Existential quantification?

Range Restriction Algorithm

Function *rr*
Input: Formula ϕ in SRNF
Output: Subset of free variables of ϕ or $\perp$
case ϕ of

- $R(t_1, \ldots, t_n)$: $rr(\phi) =$ all variables in $t_1, \ldots, t_n$;
- $X = a$ or $a = X$: $rr(\phi) = \{X\}$;
- $\phi_1 \wedge \phi_2$: $rr(\phi) = rr(\phi_1) \cup rr(\phi_2)$;
- $\phi_1 \wedge X = Y$: $rr(\phi) =$
  $$\begin{cases} rr(\phi_1) & \text{if } \{X, Y\} \cap rr(\phi_1) = \emptyset; \\ rr(\phi_1) \cup \{X, Y\} & \text{otherwise;} \end{cases}$$
- $\phi_1 \vee \phi_2$: $rr(\phi) = rr(\phi_1) \cap rr(\phi_2)$;
- $\neg\phi_1$: $rr(\phi) = \emptyset$;
- $\exists X : \psi$: if $X \in rr(\psi)$ then $rr(\phi) = rr(\psi) \setminus \{X\}$ else return $\perp$;

Assumption: Set operations with $\perp$ always result in $\perp$.

Safe Range Queries

Definition

A Relational Calculus query $\{e_1, \ldots, e_n \mid \phi\}$ is safe range, if $rr(SRNF(\phi))$ is equal to the free variables in ϕ.

Theorem

Each safe range query is domain independent.

Theorem

Any safe range query can be written as a query of Relational Algebra, and vice versa.

Safe Range Queries

Definition

A Relational Calculus query $\{e_1, \ldots, e_n \mid \phi\}$ is safe range, if $rr(SRNF(\phi))$ is equal to the free variables in ϕ.

Theorem

Each safe range query is domain independent.

Theorem

Any safe range query can be written as a query of Relational Algebra, and vice versa.

Safe Range Queries

Definition

A Relational Calculus query $\{e_1, \ldots, e_n \mid \phi\}$ is safe range, if $rr(SRNF(\phi))$ is equal to the free variables in ϕ.

Theorem

Each safe range query is domain independent.

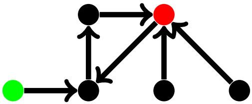
Theorem

Any safe range query can be written as a query of Relational Algebra, and vice versa.

Expressivity

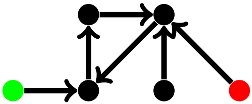
- Some simple problems cannot be represented in relational calculus.
- Example: Reachability on deterministic graphs.
- Holds also for relational algebra, SQL-92 etc.

Reachability on Deterministic Graphs



Prototypical problem for LOGSPACE!

Reachability on Deterministic Graphs



Prototypical problem for LOGSPACE!

Transitive Closure

Key notion: Transitive Closure

Definition

Given graph $G = \langle V, E \rangle$, $E \subseteq V \times V$, and $a, b \in V$, the transitive closure $TC(G) \subseteq V \times V$ is:

$$\begin{aligned} TC(G) := \; & \{(x,y) \mid (x,y) \in E\} \\ & \cup \{(x,y) \mid (x,z) \in TC(G) \wedge (z,y) \in TC(G)\} \end{aligned}$$

Note: $TC(G)$ appears in its own definition.
In relational calculus we cannot refer to what we define.

Transitive Closure

Key notion: Transitive Closure

Definition

Given graph $G = \langle V, E \rangle$, $E \subseteq V \times V$, and $a, b \in V$, the transitive closure $TC(G) \subseteq V \times V$ is:

$$\begin{aligned} TC(G) := \; & \{(x, y) \mid (x, y) \in E\} \\ & \cup \{(x, y) \mid (x, z) \in TC(G) \wedge (z, y) \in TC(G)\} \end{aligned}$$

Note: $TC(G)$ appears in its own definition.

In relational calculus we cannot refer to what we define.

Transitive Closure

Key notion: Transitive Closure

Definition

Given graph $G = \langle V, E \rangle$, $E \subseteq V \times V$, and $a, b \in V$, the transitive closure $TC(G) \subseteq V \times V$ is:

$$\begin{aligned} TC(G) := \ & \{(x, y) \mid (x, y) \in E\} \\ & \cup \{(x, y) \mid (x, z) \in TC(G) \wedge (z, y) \in TC(G)\} \end{aligned}$$

Note: $TC(G)$ appears in its own definition.
In relational calculus we cannot refer to what we define.

- Idea: Use Horn clauses for named definitions.
- It is then possible to write definitions using the concept being defined.
- Positive Datalog

Language Elements

- Set of extensional predicate symbols **PS**
- Each predicate symbol has an associated arity $ar : \mathbf{PS} \rightarrow \mathbb{N}_0$
- Set of constant symbols **CS**
- Set of variable symbols **VS**

Syntax

A Datalog rule is of the form:

$$r_1(t_{1_1}, \ldots, t_{n_1}) \leftarrow r_2(t_{1_2}, \ldots, t_{n_2}), \ldots, r_m(t_{1_m}, \ldots, t_{n_m}).$$

- $m \geqslant 1$
- $r_1, \ldots, r_m \in \mathbf{PS}$
- $t_{1_1}, \ldots, t_{n_m} \in \mathbf{CS} \cup \mathbf{VS}$
- $\forall i \; 1 \leqslant i \leqslant m : ar(r_i) = n_i$
- $((t_{1_1} \cup \ldots \cup t_{n_1}) \cap \mathbf{VS}) \subseteq ((t_{1_2} \cup \ldots \cup t_{n_m}) \cap \mathbf{VS})$

Syntax

A Datalog rule is of the form:

$$\{t_{1_1}, \ldots, t_{n_1} \mid \exists \ldots : r_2(t_{1_2}, \ldots, t_{n_2}) \wedge \ldots \wedge r_m(t_{1_m}, \ldots, t_{n_m})\}$$

- $m \geqslant 1$
- $r_1, \ldots, r_m \in \mathbf{PS}$
- $t_{1_1}, \ldots, t_{n_m} \in \mathbf{CS} \cup \mathbf{VS}$
- $\forall i \; 1 \leqslant i \leqslant m : ar(r_i) = n_i$
- $((t_{1_1} \cup \ldots \cup t_{n_1}) \cap \mathbf{VS}) \subseteq ((t_{1_2} \cup \ldots \cup t_{n_m}) \cap \mathbf{VS})$ Safe range!

Syntax

$$r_1(t_{1_1}, \ldots, t_{n_1}) \leftarrow r_2(t_{1_2}, \ldots, t_{n_2}), \ldots, r_m(t_{1_m}, \ldots, t_{n_m}).$$

- $H(r) = \{r_1(t_{1_1}, \ldots, t_{n_1})\}$
- $B(r) = \{r_2(t_{1_2}, \ldots, t_{n_2}), \ldots, r_m(t_{1_m}, \ldots, t_{n_m})\}$
- $V(r) = \{t_{1_1}, \ldots, t_{n_m}\} \cap \mathbf{VS}$
- $C(r) = \{t_{1_1}, \ldots, t_{n_m}\} \cap \mathbf{CS}$
- $H(r)$ is the head of r.
- $B(r)$ is the body of r.
- A Datalog program is a set of rules.

Semantics

Intuitively: For each rule r, whenever $B(r)$ is true, $H(r)$ should also be true. $B(r) = \emptyset$ is considered to be true.

Different ways for defining the semantics:

- model theory
- fixpoint theory
- proof theory

Semantics

Intuitively: For each rule r, whenever $B(r)$ is true, $H(r)$ should also be true. $B(r) = \emptyset$ is considered to be true.
Different ways for defining the semantics:

- model theory
- fixpoint theory
- proof theory

Outline

Model Theory

Definition (Herbrand Universe)

$$\mathbf{HU}(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} C(r)$$

Definition (Herbrand Base)

$$\mathbf{HB}(\mathcal{P}) = \{r(t_1, \ldots, t_n) \mid r \in \mathbf{PS}, t_1, \ldots, t_n \in \mathbf{HU}(\mathcal{P}), ar(r) = n\}$$

- $\mathbf{HU}(\mathcal{P})$: Constants of the program (active domain!)
- $\mathbf{HB}(\mathcal{P})$: Ground atoms constructable from $\mathbf{HU}(\mathcal{P})$

Model Theory

Definition (Herbrand Universe)

$$\mathbf{HU}(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} C(r)$$

Definition (Herbrand Base)

$$\mathbf{HB}(\mathcal{P}) = \{ r(t_1, \ldots, t_n) \mid r \in \mathbf{PS}, \\ t_1, \ldots, t_n \in \mathbf{HU}(\mathcal{P}), ar(r) = n \}$$

- $\mathbf{HU}(\mathcal{P})$: Constants of the program (active domain!)
- $\mathbf{HB}(\mathcal{P})$: Ground atoms constructable from $\mathbf{HU}(\mathcal{P})$

Model Theory

Definition (Herbrand Universe)

$$\mathbf{HU}(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} C(r)$$

Definition (Herbrand Base)

$$\begin{aligned}\mathbf{HB}(\mathcal{P}) = \{r(t_1, \ldots, t_n) \mid\ & r \in \mathbf{PS}, \\ & t_1, \ldots, t_n \in \mathbf{HU}(\mathcal{P}), ar(r) = n\}\end{aligned}$$

- $\mathbf{HU}(\mathcal{P})$: Constants of the program (active domain!)
- $\mathbf{HB}(\mathcal{P})$: Ground atoms constructable from $\mathbf{HU}(\mathcal{P})$

Example: Herbrand Base

Example

$$\mathcal{P}_r = \{ \quad \mathtt{arc(a,b).}$$
$$\mathtt{arc(b,c).}$$
$$\mathtt{reachable(a).}$$
$$\mathtt{reachable(Y)} \leftarrow \mathtt{arc(X,Y), reachable(X).} \}$$

$$\mathbf{HU}(\mathcal{P}_r) = \{\mathtt{a,b,c}\}$$
$$\mathbf{HB}(\mathcal{P}_r) = \{\mathtt{arc(a,a), arc(a,b), arc(a,c),}$$
$$\mathtt{arc(b,a), arc(b,b), arc(b,c),}$$
$$\mathtt{arc(c,a), arc(c,b), arc(c,c),}$$
$$\mathtt{reachable(a), reachable(b), reachable(c)}\}$$

Example: Herbrand Base

Example

$$\mathcal{P}_r = \{ \begin{array}{l} \mathtt{arc}(\mathtt{a},\mathtt{b}). \\ \mathtt{arc}(\mathtt{b},\mathtt{c}). \\ \mathtt{reachable}(\mathtt{a}). \\ \mathtt{reachable}(\mathtt{Y}) \leftarrow \mathtt{arc}(\mathtt{X},\mathtt{Y}), \mathtt{reachable}(\mathtt{X}). \} \end{array}$$

$$\begin{array}{rcl} \mathbf{HU}(\mathcal{P}_r) & = & \{\mathtt{a},\mathtt{b},\mathtt{c}\} \\ \mathbf{HB}(\mathcal{P}_r) & = & \{\mathtt{arc}(\mathtt{a},\mathtt{a}), \mathtt{arc}(\mathtt{a},\mathtt{b}), \mathtt{arc}(\mathtt{a},\mathtt{c}), \\ & & \mathtt{arc}(\mathtt{b},\mathtt{a}), \mathtt{arc}(\mathtt{b},\mathtt{b}), \mathtt{arc}(\mathtt{b},\mathtt{c}), \\ & & \mathtt{arc}(\mathtt{c},\mathtt{a}), \mathtt{arc}(\mathtt{c},\mathtt{b}), \mathtt{arc}(\mathtt{c},\mathtt{c}), \\ & & \mathtt{reachable}(\mathtt{a}), \mathtt{reachable}(\mathtt{b}), \mathtt{reachable}(\mathtt{c})\} \end{array}$$

Instantiation

Definition

Valuation $v_{\mathcal{P}}(r)$ of a rule r: Set of all substitutions $V(r) \rightarrow \mathbf{HU}(\mathcal{P})$

Definition (Instantiation of a rule r)

$Ground_{\mathcal{P}}(r) = \bigcup_{v \in v_{\mathcal{P}}(r)} v(r)$

Definition (Instantiation of a program $\mathcal{P}$)

$Ground(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} Ground_{\mathcal{P}}(r)$

Instantiation

Definition

Valuation $v_{\mathcal{P}}(r)$ of a rule r: Set of all substitutions $V(r) \rightarrow \mathbf{HU}(\mathcal{P})$

Definition (Instantiation of a rule r)

$Ground_{\mathcal{P}}(r) = \bigcup_{v \in v_{\mathcal{P}}(r)} v(r)$

Definition (Instantiation of a program $\mathcal{P}$)

$Ground(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} Ground_{\mathcal{P}}(r)$

Instantiation

Definition

Valuation $v_{\mathcal{P}}(r)$ of a rule r: Set of all substitutions $V(r) \rightarrow \mathbf{HU}(\mathcal{P})$

Definition (Instantiation of a rule r)

$Ground_{\mathcal{P}}(r) = \bigcup_{v \in v_{\mathcal{P}}(r)} v(r)$

Definition (Instantiation of a program $\mathcal{P}$)

$Ground(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} Ground_{\mathcal{P}}(r)$

Example: Instantiation

Example

$$\mathcal{P}_r = \{ \quad \texttt{arc(a,b). arc(b,c). reachable(a).}$$
$$\texttt{reachable(Y)} \leftarrow \texttt{arc(X,Y), reachable(X).} \}$$

$$\begin{aligned} \textit{Ground}(\mathcal{P}_r) = \{ & \texttt{arc(a,b). arc(b,c). reachable(a).} \\ & \texttt{reachable(a)} \leftarrow \texttt{arc(a,a), reachable(a).} \\ & \texttt{reachable(b)} \leftarrow \texttt{arc(a,b), reachable(a).} \\ & \texttt{reachable(c)} \leftarrow \texttt{arc(a,c), reachable(a).} \\ & \texttt{reachable(a)} \leftarrow \texttt{arc(b,a), reachable(b).} \\ & \texttt{reachable(b)} \leftarrow \texttt{arc(b,b), reachable(b).} \\ & \texttt{reachable(c)} \leftarrow \texttt{arc(b,c), reachable(b).} \\ & \texttt{reachable(a)} \leftarrow \texttt{arc(c,a), reachable(c).} \\ & \texttt{reachable(b)} \leftarrow \texttt{arc(c,b), reachable(c).} \\ & \texttt{reachable(c)} \leftarrow \texttt{arc(c,c), reachable(c).} \} \end{aligned}$$

Herbrand Models

Definition ((Herbrand-) Interpretations I for $\mathcal{P}$)

$I \subseteq \mathbf{HB}(\mathcal{P})$

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (H(r) \subseteq M) \vee (B(r) \not\subseteq M)$

"If the body is true, the head must be true."

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (B(r) \subseteq M) \rightarrow (H(r) \subseteq M)$

Herbrand Models

Definition ((Herbrand-) Interpretations I for $\mathcal{P}$)

$I \subseteq \mathbf{HB}(\mathcal{P})$

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (H(r) \subseteq M) \vee (B(r) \nsubseteq M)$

"If the body is true, the head must be true."

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (B(r) \subseteq M) \rightarrow (H(r) \subseteq M)$

Herbrand Models

Definition ((Herbrand-) Interpretations I for $\mathcal{P}$)

$I \subseteq \mathbf{HB}(\mathcal{P})$

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (H(r) \subseteq M) \vee (B(r) \nsubseteq M)$

"If the body is true, the head must be true."

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (B(r) \subseteq M) \rightarrow (H(r) \subseteq M)$

Herbrand Models

Definition ((Herbrand-) Interpretations I for $\mathcal{P}$)

$I \subseteq \mathbf{HB}(\mathcal{P})$

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (H(r) \subseteq M) \lor (B(r) \nsubseteq M)$

"If the body is true, the head must be true."

Definition ((Herbrand-) Models for $\mathcal{P}$)

$M \subseteq \mathbf{HB}(\mathcal{P})$ such that
$\forall r \in Ground(\mathcal{P}) : (B(r) \subseteq M) \rightarrow (H(r) \subseteq M)$

Example: Herbrand Models

Example

$$\mathcal{P}_r = \{ \quad \mathtt{arc(a,b).\ arc(b,c).\ reachable(a).} \\ \mathtt{reachable(Y)} \leftarrow \mathtt{arc(X,Y), reachable(X).}\ \}$$

$$M_1 = \{ \quad \mathtt{arc(a,b), arc(b,c),} \\ \mathtt{reachable(a), reachable(b), reachable(c)}\}$$

$$M_2 = \mathbf{HB}(\mathcal{P}_r)$$

All $M : M_1 \subseteq M \subseteq M_2$ are models and only these.

Example: Herbrand Models

Example

$$\mathcal{P}_r = \{ \quad \texttt{arc(a,b). arc(b,c). reachable(a).}$$
$$\texttt{reachable(Y)} \leftarrow \texttt{arc(X,Y), reachable(X).} \}$$
$$M_1 = \{ \quad \texttt{arc(a,b), arc(b,c),}$$
$$\texttt{reachable(a), reachable(b), reachable(c)}\}$$
$$M_2 = \mathbf{HB}(\mathcal{P}_r)$$

All $M : M_1 \subseteq M \subseteq M_2$ are models and only these.

Example: Herbrand Models

Example

$$\mathcal{P}_r = \{ \quad \mathtt{arc(a,b).\ arc(b,c).\ reachable(a).}$$
$$\mathtt{reachable(Y)} \leftarrow \mathtt{arc(X,Y), reachable(X).}\ \}$$

$$M_1 = \{ \quad \mathtt{arc(a,b), arc(b,c),}$$
$$\mathtt{reachable(a), reachable(b), reachable(c)}\}$$

$$M_2 = \mathbf{HB}(\mathcal{P}_r)$$

All $M : M_1 \subseteq M \subseteq M_2$ are models and only these.

Example: Herbrand Models

Example

$$\mathcal{P}_r = \{ \quad \mathtt{arc(a,b).\ arc(b,c).\ reachable(a).} \\ \mathtt{reachable(Y)} \leftarrow \mathtt{arc(X,Y), reachable(X).}\ \}$$

$$M_1 = \{ \quad \mathtt{arc(a,b), arc(b,c),} \\ \mathtt{reachable(a), reachable(b), reachable(c)}\}$$

$$M_2 = \mathbf{HB}(\mathcal{P}_r)$$

All $M : M_1 \subseteq M \subseteq M_2$ are models and only these.

Example: Herbrand Models

Example

$$\mathcal{P}_r = \{ \quad \mathtt{arc(a,b).\ arc(b,c).\ reachable(a).} \\ \mathtt{reachable(Y)} \leftarrow \mathtt{arc(X,Y), reachable(X).} \}$$

$$M_1 = \{ \quad \mathtt{arc(a,b), arc(b,c),} \\ \mathtt{reachable(a), reachable(b), reachable(c)} \}$$

$$M_2 = \mathbf{HB}(\mathcal{P}_r)$$

All $M : M_1 \subseteq M \subseteq M_2$ are models and only these.

Minimal Models

Theorem

HB$(\mathcal{P})$ *is always a model for any Datalog program* $\mathcal{P}$*.*

Theorem

Each Datalog program $\mathcal{P}$ *has a unique subset minimal model* $MM(\mathcal{P})$*.*

Definition

The semantics of a Datalog program $\mathcal{P}$ is given by $MM(\mathcal{P})$

Note: Each element of $MM(\mathcal{P})$ is a logical consequence of $\mathcal{P}$.

Minimal Models

Theorem

HB$(\mathcal{P})$ *is always a model for any Datalog program* $\mathcal{P}$*.*

Theorem

Each Datalog program $\mathcal{P}$ *has a unique subset minimal model* *MM*$(\mathcal{P})$*.*

Definition

The semantics of a Datalog program $\mathcal{P}$ is given by *MM*$(\mathcal{P})$

Note: Each element of *MM*$(\mathcal{P})$ is a logical consequence of $\mathcal{P}$.

Minimal Models

Theorem

***HB**$(\mathcal{P})$ is always a model for any Datalog program $\mathcal{P}$.*

Theorem

Each Datalog program $\mathcal{P}$ has a unique subset minimal model $MM(\mathcal{P})$.

Definition

The semantics of a Datalog program $\mathcal{P}$ is given by $MM(\mathcal{P})$

Note: Each element of $MM(\mathcal{P})$ is a logical consequence of $\mathcal{P}$.

Minimal Models

Theorem

HB$(\mathcal{P})$ *is always a model for any Datalog program* $\mathcal{P}$*.*

Theorem

Each Datalog program $\mathcal{P}$ *has a unique subset minimal model* *MM*$(\mathcal{P})$*.*

Definition

The semantics of a Datalog program $\mathcal{P}$ is given by *MM*$(\mathcal{P})$

Note: Each element of *MM*$(\mathcal{P})$ is a logical consequence of $\mathcal{P}$.

Outline

Concept: Operator

"If we assume that all atoms in I are true, which other atoms must be true in order to satisfy the program?"

- Start with $I = \emptyset$ (nothing is true).
- Define operator $\mathbf{T}_{\mathcal{P}}$.
- Apply $\mathbf{T}_{\mathcal{P}}$, until there are no further additions.
- The obtained result (fixpoint) defines the semantics.

Immediate Consequences

Definition (Operator $\mathbf{T}_{\mathcal{P}}$ for Datalog program $\mathcal{P}$)

Given an interpretation I,

$$\mathbf{T}_{\mathcal{P}}(I) = \{h \mid r \in Ground(\mathcal{P}), B(r) \subseteq I, h \in H(r)\}$$

- $\mathbf{T}_{\mathcal{P}}(I)$ extends I, such that unsatisfied rules (w.r.t. I) become satisfied.
- Other rules may become unsatisfied w.r.t. $\mathbf{T}_{\mathcal{P}}(I)$.
- $\Rightarrow$ Iterative application.

Example: Immediate Consequences

Example

$$\mathcal{P}_r = \{ \quad \mathtt{arc(a,b).\ arc(b,c).\ reachable(a).}$$
$$\mathtt{reachable(Y)} \leftarrow \mathtt{arc(X,Y), reachable(X).}\ \}$$

1. $\mathbf{T}_{\mathcal{P}_r}(\emptyset) = \{\mathtt{arc(a,b),\ arc(b,c),\ reachable(a)}\}$
2. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) = \mathbf{T}_{\mathcal{P}_r}(\emptyset) \cup \{\mathtt{reachable(b)}\}$
3. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) \cup \{\mathtt{reachable(c)}\}$
4. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))$
5. $\{\mathtt{arc(a,b),\ arc(b,c),}$
   $\mathtt{reachable(a),\ reachable(b),\ reachable(c)}\}$

Example: Immediate Consequences

Example

$$\mathcal{P}_r = \{ \quad \mathtt{arc(a,b).\ arc(b,c).\ reachable(a).} \\ \mathtt{reachable(Y)} \leftarrow \mathtt{arc(X,Y), reachable(X).}\ \}$$

1. $\mathbf{T}_{\mathcal{P}_r}(\emptyset) = \{\mathtt{arc(a,b),\ arc(b,c),\ reachable(a)}\}$
2. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) = \mathbf{T}_{\mathcal{P}_r}(\emptyset) \cup \{\mathtt{reachable(b)}\}$
3. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) \cup \{\mathtt{reachable(c)}\}$
4. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))$
5. $\{\mathtt{arc(a,b),\ arc(b,c),}$ $\mathtt{reachable(a),\ reachable(b),\ reachable(c)}\}$

Example: Immediate Consequences

Example

$$\mathcal{P}_r = \{ \quad \mathtt{arc(a,b).\ arc(b,c).\ reachable(a).} \\ \quad \mathtt{reachable(Y)} \leftarrow \mathtt{arc(X,Y), reachable(X).}\ \}$$

1. $\mathbf{T}_{\mathcal{P}_r}(\emptyset) = \{\mathtt{arc(a,b)},\ \mathtt{arc(b,c)},\ \mathtt{reachable(a)}\}$
2. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) = \mathbf{T}_{\mathcal{P}_r}(\emptyset) \cup \{\mathtt{reachable(b)}\}$
3. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) \cup \{\mathtt{reachable(c)}\}$
4. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))$
5. $\{\mathtt{arc(a,b)},\ \mathtt{arc(b,c)},\ \mathtt{reachable(a)},\ \mathtt{reachable(b)},\ \mathtt{reachable(c)}\}$

Example: Immediate Consequences

Example

$$\mathcal{P}_r = \{ \quad \texttt{arc(a,b). arc(b,c). reachable(a).} \\ \texttt{reachable(Y)} \leftarrow \texttt{arc(X,Y), reachable(X).} \}$$

1. $\mathbf{T}_{\mathcal{P}_r}(\emptyset) = \{\texttt{arc(a,b), arc(b,c), reachable(a)}\}$
2. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) = \mathbf{T}_{\mathcal{P}_r}(\emptyset) \cup \{\texttt{reachable(b)}\}$
3. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) \cup \{\texttt{reachable(c)}\}$
4. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))$
5. $\{\texttt{arc(a,b), arc(b,c),}$ $\texttt{reachable(a), reachable(b), reachable(c)}\}$

Example: Immediate Consequences

Example

$$\mathcal{P}_r = \{ \quad \texttt{arc(a,b). arc(b,c). reachable(a).}$$
$$\texttt{reachable(Y)} \leftarrow \texttt{arc(X,Y), reachable(X).} \}$$

1. $\mathbf{T}_{\mathcal{P}_r}(\emptyset) = \{\texttt{arc(a,b), arc(b,c), reachable(a)}\}$
2. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) = \mathbf{T}_{\mathcal{P}_r}(\emptyset) \cup \{\texttt{reachable(b)}\}$
3. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)) \cup \{\texttt{reachable(c)}\}$
4. $\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))) = \mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\mathbf{T}_{\mathcal{P}_r}(\emptyset)))$
5. $\{\texttt{arc(a,b), arc(b,c),}$
   $\texttt{reachable(a), reachable(b), reachable(c)}\}$

Properties of $\mathbf{T}_{\mathcal{P}}$

Lattice: $V = (P(\mathbf{HB}(\mathcal{P})), \subseteq)$

$\forall X \subseteq V : \exists inf(X) \wedge \exists sup(X)$

$inf(V) = \emptyset$, $sup(V) = \mathbf{HB}(\mathcal{P})$

Monotony: $X \subseteq Y \rightarrow \mathbf{T}_{\mathcal{P}}(X) \subseteq \mathbf{T}_{\mathcal{P}}(Y)$

Continuity: $\forall X \subseteq V : \mathbf{T}_{\mathcal{P}}(sup(X)) = sup(\mathbf{T}_{\mathcal{P}}(X))$

Properties of $\mathbf{T}_{\mathcal{P}}$

Lattice: $V = (P(\mathbf{HB}(\mathcal{P})), \subseteq)$

$\forall X \subseteq V : \exists inf(X) \wedge \exists sup(X)$

$inf(V) = \emptyset$, $sup(V) = \mathbf{HB}(\mathcal{P})$

Monotony: $X \subseteq Y \rightarrow \mathbf{T}_{\mathcal{P}}(X) \subseteq \mathbf{T}_{\mathcal{P}}(Y)$

Continuity: $\forall X \subseteq V : \mathbf{T}_{\mathcal{P}}(sup(X)) = sup(\mathbf{T}_{\mathcal{P}}(X))$

Properties of $\mathbf{T}_{\mathcal{P}}$

Lattice: $V = (P(\mathbf{HB}(\mathcal{P})), \subseteq)$

$\forall X \subseteq V : \exists inf(X) \wedge \exists sup(X)$

$inf(V) = \emptyset$, $sup(V) = \mathbf{HB}(\mathcal{P})$

Monotony: $X \subseteq Y \rightarrow \mathbf{T}_{\mathcal{P}}(X) \subseteq \mathbf{T}_{\mathcal{P}}(Y)$

Continuity: $\forall X \subseteq V : \mathbf{T}_{\mathcal{P}}(sup(X)) = sup(\mathbf{T}_{\mathcal{P}}(X))$

Properties of $\mathbf{T}_{\mathcal{P}}$

Lattice: $V = (P(\mathbf{HB}(\mathcal{P})), \subseteq)$

$\forall X \subseteq V : \exists inf(X) \wedge \exists sup(X)$

$inf(V) = \emptyset$, $sup(V) = \mathbf{HB}(\mathcal{P})$

Monotony: $X \subseteq Y \rightarrow \mathbf{T}_{\mathcal{P}}(X) \subseteq \mathbf{T}_{\mathcal{P}}(Y)$

Continuity: $\forall X \subseteq V : \mathbf{T}_{\mathcal{P}}(sup(X)) = sup(\mathbf{T}_{\mathcal{P}}(X))$

Properties of $\mathbf{T}_{\mathcal{P}}$

Lattice: $V = (P(\mathbf{HB}(\mathcal{P})), \subseteq)$

$\forall X \subseteq V : \exists inf(X) \wedge \exists sup(X)$

$inf(V) = \emptyset$, $sup(V) = \mathbf{HB}(\mathcal{P})$

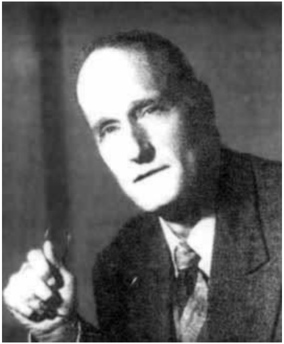
Monotony: $X \subseteq Y \rightarrow \mathbf{T}_{\mathcal{P}}(X) \subseteq \mathbf{T}_{\mathcal{P}}(Y)$

Continuity: $\forall X \subseteq V : \mathbf{T}_{\mathcal{P}}(sup(X)) = sup(\mathbf{T}_{\mathcal{P}}(X))$

Knaster, Tarski, Kleene

Bronisław Knaster
(1893–1990)

Alfred Tarski
(1902–1983)



Stephen Kleene
(1909–1994)

Existence of Fixpoints

Theorem

$\mathbf{T}_{\mathcal{P}}$ *is monotone and continuous on the lattice of interpretations and subset relations.*

Theorem (Knaster-Tarski)

For monotone operators on lattices a least fixpoint exists, and it is inf$(\{X \mid \mathbf{T}_{\mathcal{P}}(X) \subseteq X\})$

Construction of Fixpoints

Theorem (Kleene)

For continuous operators on lattices the least fixpoint can be computed by iteration starting from the infimum.
$\mathbf{T}_{\mathcal{P}}^{\omega} = sup(\{\mathbf{T}_{\mathcal{P}}^{i} \mid i \geqslant 0\})$,
$\mathbf{T}_{\mathcal{P}}^{0} = inf(V)$, $\mathbf{T}_{\mathcal{P}}^{i} = \mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}^{i-1})$

Corollary

Our lattice is finite, therefore the least fixpoint of $\mathbf{T}_{\mathcal{P}}$ can be computed by a finite number of iterations starting from $\emptyset$.

$\mathbf{T}_{\mathcal{P}}^{\omega}$ – Minimal Model

Theorem

For all Datalog programs $\mathcal{P}$, we can show $\mathbf{T}_{\mathcal{P}}^{\omega} = MM(\mathcal{P})$.

Note: All consequences of a program can be computed by iteration over the immediate consequences.

$\mathbf{T}_{\mathcal{P}}^{\omega}$ – Minimal Model

Theorem

For all Datalog programs $\mathcal{P}$, we can show $\mathbf{T}_{\mathcal{P}}^{\omega} = MM(\mathcal{P})$.

Note: All consequences of a program can be computed by iteration over the immediate consequences.

Outline

Reminder: Horn and Goal Clauses, SLD Resolution

- A Horn clause is a clause containing at most one positive literal.
- A Goal clause is a clause containing no positive literal.
- SLD Resolution: Linear resolution, where at each step only goal clauses and (instances of) input clauses are used.

Theorem

SLD resolution is refutation complete for Horn clauses.

SLD Resolution for Datalog

- View rules Horn clauses
- Apply SLD Resolution
- Unification is simple – absence of function symbols.

Definition (SLD Resolution Semantics)

Let $SLD(\mathcal{P})$ denote the set of ground atoms, for which an SLD refutation w.r.t. $\mathcal{P}$ exists.

Equivalence

Theorem

For all Datalog programs $\mathcal{P}$, we can show $SLD(\mathcal{P}) = \mathbf{T}_{\mathcal{P}}^{\omega} = MM(\mathcal{P})$.

Nonmonotonic Queries

- Some simple queries cannot be written in positive Datalog.
- Example: $(\pi_1\ R) - S$
- This query is nonmonotone!
- Adding tuples to S may retract result tuples.
- Positive Datalog can express only monotone queries.

Nonmonotonic Queries

- In Relational Calculus $(\pi_1\ R) - S$ is written using negation.
- Introduce negation also for Datalog!
- Problem: Negation through recursion?

Outline

Closed World Assumption

- Atoms for which it is not necessary to be true should be considered as false.
- Only those items which are known should be true.
- Example: Timetable
- Reason for Minimal Model semantics!

Closed World Assumption

Definition

For a positive program $\mathcal{P}$, $CWA(\mathcal{P}) = \{A \mid \mathcal{P} \not\models A\}$.
Equivalently: $CWA(\mathcal{P}) = \mathbf{HB}(\mathcal{P}) - MM(\mathcal{P})$

Is this as simple if we allow rules with negative body literals?

Normal Programs – Syntax

Definition

A normal rule is

$$h \leftarrow b_1, \ldots, b_m, \mathtt{not}\ b_{m+1}, \ldots, \mathtt{not}\ b_n.$$
$$1 \leqslant m \leqslant n$$

Let

$B^+(r) = \{b_1, \ldots, b_m\}$

$B^-(r) = \{b_{m+1}, \ldots, b_m\}$

$\mathtt{not}.a = \mathtt{not}\ a, \mathtt{not}.\mathtt{not}\ a = a$

$\mathtt{not}.L = \{\mathtt{not}.l \mid l \in L\}$

$B(r) = B^+(r) \cup \mathtt{not}.B^-(r)$

$H(r), V(r), C(r)$ as before

Unsafe Queries

Recall: Using Negation it is easy to violate domain independence!

Example

$$positive(X) \leftarrow \texttt{not}\ zero(X).$$

Definition (Safety)

Each variable in a rule must occur in a positive body atom.

Example

$$positive(X) \leftarrow number(X), \texttt{not}\ zero(X).$$

Unsafe Queries

Recall: Using Negation it is easy to violate domain independence!

Example

$$\mathit{positive}(X) \leftarrow \mathtt{not}\ \mathit{zero}(X).$$

Definition (Safety)

Each variable in a rule must occur in a positive body atom.

Example

$$\mathit{positive}(X) \leftarrow \mathit{number}(X), \mathtt{not}\ \mathit{zero}(X).$$

Normal Programs – Semantics

- Most concepts do not change.
- Satisfaction of a rule r with respect to M:
  If $B^+(r) \subseteq M$ and $M \cap B^-(r) = \emptyset$, then $H(r) \in M$
- Question: Minimal Model semantics suitable?

Normal Programs

In general there is no unique minimal model.

Example

$$a \leftarrow \texttt{not}\ b.$$

There are two models $M_1 = \{a\}$ and $M_2 = \{b\}$.
M_2 is not very intuitive.

Normal Programs

In general there is no unique minimal model.

Example

$$a \leftarrow \texttt{not}\ b.$$

There are two models $M_1 = \{a\}$ and $M_2 = \{b\}$.
M_2 is not very intuitive.

Normal Programs

In general there is no unique minimal model.

Example

$$a \leftarrow \texttt{not } b.$$

There are two models $M_1 = \{a\}$ and $M_2 = \{b\}$.
M_2 is not very intuitive.

Normal Programs

Semantics of "negative recursion"?

$$\begin{aligned}&\mathit{person}(\mathit{nicola}).\\&\mathit{male}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{female}(X).\\&\mathit{female}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{male}(X).\end{aligned}$$

$\{\mathit{person}(\mathit{nicola}), \mathit{male}(\mathit{nicola})\}$ and
$\{\mathit{person}(\mathit{nicola}), \mathit{female}(\mathit{nicola})\}$ are minimal models

Both are equally intuitive.

Normal Programs

Semantics of “negative recursion”?

$$\begin{array}{l} person(nicola). \\ male(X) \leftarrow person(X), \texttt{not}\ female(X). \\ female(X) \leftarrow person(X), \texttt{not}\ male(X). \end{array}$$

$\{person(nicola), male(nicola)\}$ and
$\{person(nicola), female(nicola)\}$ are minimal models

Both are equally intuitive.

Normal Programs

Semantics of “negative recursion”?

$$
\begin{aligned}
&\mathit{person}(\mathit{nicola}).\\
&\mathit{male}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{female}(X).\\
&\mathit{female}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{male}(X).
\end{aligned}
$$

$\{\mathit{person}(\mathit{nicola}), \mathit{male}(\mathit{nicola})\}$ and
$\{\mathit{person}(\mathit{nicola}), \mathit{female}(\mathit{nicola})\}$ are minimal models

Both are equally intuitive.

Possibilities

1. Pragmatic: Do not allow “recursion through negation”.
2. Three-valued: Stay with a unique model, which may leave some atoms undefined.
3. Two-valued: Abandon model uniqueness, stay with standard models.

Outline

Dependency Graph

Definition

For a negative Datalog program $\mathcal{P}$, we define a directed graph (V, E), where V are the predicate symbols of $\mathcal{P}$, and $(p, q) \in E$ if p is in the head and q is in the body of some rule. If q is in the negative body, we mark the arc.

Examples

Example

$$a \leftarrow b.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Example

$$a \leftarrow b, c.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Examples

Example

$$a \leftarrow b.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Example

$$a \leftarrow b, c.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Stratification

Main idea: Partition the program along negation.

Definition

A stratification is a function λ, which maps predicate symbols to integers such that for each rule with *p* being the head predicate the following conditions hold:

1. For each predicate *q* in the positive body, $\lambda(p) \geqslant \lambda(q)$.
2. For each predicate *r* in the negative body, $\lambda(p) > \lambda(r)$.

Stratification

- λ induces a partition $\langle P_0, \ldots, P_n \rangle$ of $\mathcal{P}$ (assuming that λ maps to integers between 0 and n):

$$
\begin{aligned}
P_0 &= \{r \mid \lambda(H(r)) = 0\} \\
\ldots \\
P_n &= \{r \mid \lambda(H(r)) = n\}
\end{aligned}
$$

- λ defines a partial ordering between partitions.
- We can evaluate the program along this ordering.

Examples

Example

$$a \leftarrow b.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Stratifiable: $\lambda(a) = 0, \lambda(b) = 0, \lambda(c) = 1$

Example

$$a \leftarrow b, c.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Not stratifiable: $\lambda(c) > \lambda(b) \geqslant \lambda(a) \geqslant \lambda(c)$

Examples

Example

$$a \leftarrow b.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Stratifiable: $\lambda(a) = 0, \lambda(b) = 0, \lambda(c) = 1$

Example

$$a \leftarrow b, c.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Not stratifiable: $\lambda(c) > \lambda(b) \geqslant \lambda(a) \geqslant \lambda(c)$

Examples

Example

$$a \leftarrow b.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Stratifiable: $\lambda(a) = 0, \lambda(b) = 0, \lambda(c) = 1$

Example

$$a \leftarrow b, c.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Not stratifiable: $\lambda(c) > \lambda(b) \geqslant \lambda(a) \geqslant \lambda(c)$

Examples

Example

$$a \leftarrow b.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Stratifiable: $\lambda(a) = 0, \lambda(b) = 0, \lambda(c) = 1$

Example

$$a \leftarrow b, c.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Not stratifiable: $\lambda(c) > \lambda(b) \geqslant \lambda(a) \geqslant \lambda(c)$

Stratification

Theorem

A program is stratifiable if and only if its dependency graph contains no cycle with a marked ("negative") edge.

Perfect Models

- Stratification specifies an order for evaluation.
- First fully compute the relations in the lowest stratum.
- Then move one stratum up and evaluate the relations there.
- Negation is evaluated only over fully computed relations.
- Can be treated like negation over EDB predicates.

Perfect Models and $\mathbf{T}_{\mathcal{P}}$

Modify operator $\mathbf{T}_{\mathcal{P}}$, as $\mathcal{P}$ may contain negation.

Definition

$$\mathbf{T}_{\mathcal{P}}(I) = \{h \mid r \in Ground(\mathcal{P}), B^+(r) \subseteq I, h \in H(r), \\ \mathtt{not}.B^-(r) \cap I = \emptyset\} \cup I$$

Perfect Models and $\mathbf{T}_{\mathcal{P}}$

Definition

Let $\langle P_0, \ldots, P_n \rangle$ be the partitions of a stratifiable program $\mathcal{P}$, induced by a stratification λ.
The sequence $M_0 = \mathbf{T}_{P_0}^{\infty}(\emptyset)$, $M_1 = \mathbf{T}_{P_1}^{\infty}(M_0)$, $\ldots$,
$M_n = \mathbf{T}_{P_n}^{\infty}(M_{n-1})$ defines the Perfect Model M_n of $\mathcal{P}$.

Example – stratifiable

Easy case: Negation only on EDB predicates

Example

$$color(yellow, k1).\ color(yellow, k2).\ color(blue, k3).$$
$$color(green, k4).\ color(red, k5).$$

$$block(K) \leftarrow color(F, K). \quad block(K) \leftarrow form(F, K).$$
$$diffcolor(K1, K2) \leftarrow$$
$$\quad color(F, K1), block(K2), \mathtt{not}\ color(F, K2).$$

Example – stratifiable

Example

$$form(box, k1).\ form(cone, k2).\ form(disc, k3).$$
$$form(box, k4).\ form(pyramid, k5).$$

$$block(K) \leftarrow color(F, K). \quad block(K) \leftarrow form(F, K).$$
$$pointy\_top(K) \leftarrow block(K), form(cone, K).$$
$$pointy\_top(K) \leftarrow block(K), form(pyramid, K).$$
$$fits\_on(K1, K2) \leftarrow block(K1), block(K2), \texttt{not}\ pointy\_top(K2).$$

Example – stratifiable

Example

$$form(box, k1).\ form(cone, k2).\ form(disc, k3).$$
$$form(box, k4).\ form(pyramid, k5).$$

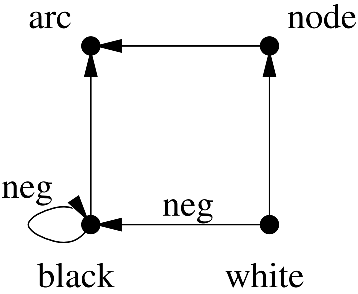
$$block(K) \leftarrow color(F, K). \quad block(K) \leftarrow form(F, K).$$
$$flat\_top(K) \leftarrow block(K), form(box, K).$$
$$flat\_top(K) \leftarrow block(K), form(disc, K).$$
$$pointy\_top(K) \leftarrow block(K), \texttt{not}\ flat\_top(K).$$
$$fits\_on(K1, K2) \leftarrow block(K1), block(K2), \texttt{not}\ pointy\_top(K2).$$

Example – unstratified

$$
\begin{array}{l}
arc(a,b).\ arc(b,c).\ arc(b,d).\\
node(N) \leftarrow arc(N,Y).\ node(N) \leftarrow arc(X,N).\\
black(Y) \leftarrow arc(X,Y), \mathtt{not}\ black(X).\\
white(X) \leftarrow node(X), \mathtt{not}\ black(X).
\end{array}
$$

Example – unstratified

Dependency Graph:



Perfect Models

- Note: Perfect Models are defined only on stratifiable programs.

Theorem

For any stratifiable program, there exists a unique Perfect Model.

Unstratifiable Programs

Example

$$
\begin{aligned}
&\mathit{person}(\mathit{nicola}).\\
&\mathit{alive}(X) \leftarrow \mathit{person}(X).\\
&\mathit{male}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{female}(X).\\
&\mathit{female}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{male}(X).
\end{aligned}
$$

Perfect Models are not defined.
But we would like to conclude at least $\mathit{alive}(\mathit{nicola})$.

Unstratifiable Programs

Example

$$\begin{aligned}
&\mathit{person}(\mathit{nicola}).\\
&\mathit{alive}(X) \leftarrow \mathit{person}(X).\\
&\mathit{male}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{female}(X).\\
&\mathit{female}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{male}(X).
\end{aligned}$$

Perfect Models are not defined.

But we would like to conclude at least $\mathit{alive}(\mathit{nicola})$.

Unstratifiable Programs

Example

$$
\begin{aligned}
&\mathit{person}(\mathit{nicola}).\\
&\mathit{alive}(X) \leftarrow \mathit{person}(X).\\
&\mathit{male}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{female}(X).\\
&\mathit{female}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{male}(X).
\end{aligned}
$$

Perfect Models are not defined.
But we would like to conclude at least $\mathit{alive}(\mathit{nicola})$.

Outline

Recursive Negation

Example

$$
\begin{array}{l}
\mathit{person}(\mathit{nicola}).\\
\mathit{alive}(X) \leftarrow \mathit{person}(X).\\
\mathit{male}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{female}(X).\\
\mathit{female}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{male}(X).
\end{array}
$$

Recursive Negation

Example

Using generalized $\mathbf{T}_{\mathcal{P}}$:

$$
\begin{array}{l}
\mathbf{T}_{\mathcal{P}}(\emptyset) = \{person(nicola)\} \\
\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\emptyset)) = \{person(nicola), alive(nicola), male(nicola), female(nicola)\} \\
\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\emptyset))) = \{person(nicola), alive(nicola)\} \\
\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\emptyset)))) = \mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\emptyset)) \\
\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\emptyset))))) = \mathbf{T}_{\mathcal{P}}(\emptyset) \\
\ldots
\end{array}
$$

Recursive Negation

Example

But there are two fixpoints:

$$\begin{aligned}
\mathbf{T}_{\mathcal{P}}(\{\mathit{person}(\mathit{nicola}), \mathit{alive}(\mathit{nicola}), \mathit{male}(\mathit{nicola})\}) = \\
\{\mathit{person}(\mathit{nicola}), \mathit{alive}(\mathit{nicola}), \mathit{male}(\mathit{nicola})\} \\
\mathbf{T}_{\mathcal{P}}(\{\mathit{person}(\mathit{nicola}), \mathit{alive}(\mathit{nicola}), \mathit{female}(\mathit{nicola})\}) = \\
\{\mathit{person}(\mathit{nicola}), \mathit{alive}(\mathit{nicola}), \mathit{female}(\mathit{nicola})\}
\end{aligned}$$

Recursive Negation

Two ways of resolving this:

1. Be cautious and do not say anything about $male(nicola)$ and $female(nicola)$.
2. Consider two scenarios: One in which $male(nicola)$ is true, another in which $female(nicola)$ is true.

Problems to resolve:

1. needs another truth value undefined.
2. allows more than one model.

Recursive Negation

Two ways of resolving this:

1. Be cautious and do not say anything about $male(nicola)$ and $female(nicola)$.
2. Consider two scenarios: One in which $male(nicola)$ is true, another in which $female(nicola)$ is true.

Problems to resolve:

1. needs another truth value undefined.
2. allows more than one model.

Outline

Three-valued Interpretations

Definition

A three-valued (or partial) interpretation I is a set of ground $\texttt{not}$ literals, such that for any ground atom a not both $a \in I$ and $\texttt{not}\, a \in I$.

Example

$I = \{\texttt{not}\ a, c\}$

- a is false in I
- b is undefined in I
- c is true in I

Three-valued Interpretations

Definition

A three-valued (or partial) interpretation I is a set of ground $\mathtt{not}$ literals, such that for any ground atom a not both $a \in I$ and $\mathtt{not}\, a \in I$.

Example

$I = \{\mathtt{not}\ a, c\}$

- a is false in I
- b is undefined in I
- c is true in I

Three-valued Interpretations

Definition

A three-valued (or partial) interpretation I is a set of ground $\mathtt{not}$ literals, such that for any ground atom a not both $a \in I$ and $\mathtt{not}\, a \in I$.

Example

$I = \{\mathtt{not}\ a, c\}$

- a is false in I
- b is undefined in I
- c is true in I

Unfounded Sets

Goal: Derive as much negative information as possible.

Example

$$a \leftarrow \texttt{not}\ b.$$

b does not occur in any head, thus can never become true and should be false. *a* should therefore be true.

Unfounded Sets

Goal: Derive as much negative information as possible.

Example

$$a \leftarrow b.$$
$$c \leftarrow \texttt{not } a.$$

Given the interpretation $\{\texttt{not } b\}$, a can never become true and should be false. c should be true in this case.

Unfounded Sets

Goal: Derive as much negative information as possible.

Example

$$a \leftarrow b.$$
$$b \leftarrow a.$$
$$c \leftarrow \texttt{not}\ a.$$

a and b occur in some heads, but all bodies of these rules require one of a or b to become true. Therefore a and b can become true only via themselves and should be false, hence c should be true.

Unfounded Sets – Definition

Definition

A set $U \subseteq \mathbf{HB}(\mathcal{P})$ is unfounded with respect to a partial interpretation I if the following holds:
For each $a \in U$ and each rule $r \in Ground(\mathcal{P})$ with $H(r) = \{a\}$ at least one of the the following conditions holds:

1. $\exists \ell \in B(r) : \mathtt{not}.\ell \in I$
2. $B^+(r) \cap U \neq \emptyset$

Unfounded Sets – Example

Example

$$a \leftarrow \texttt{not}\ b.$$

For $I = \emptyset$, $\{b\}$ is an unfounded set.

Unfounded Sets – Example

Example

$$a \leftarrow b.$$
$$c \leftarrow \mathtt{not}\ a.$$

For $I = \{\mathtt{not}\ b\}$, $\{a\}$ is an unfounded set.

Unfounded Sets – Example

Example

$$a \leftarrow b.$$
$$b \leftarrow a.$$
$$c \leftarrow \texttt{not}\ a.$$

For $I = \emptyset$, $\{a, b\}$ is an unfounded set, because condition 2 holds for $a \leftarrow b.$ and $b \leftarrow a.$.

Unfounded Operator

Theorem

For any program $\mathcal{P}$ and partial interpretation I, the greatest unfounded set $GUS_{\mathcal{P}}(I)$ (which is a superset of all unfounded sets) exists and is unique.

Idea: Use $GUS_{\mathcal{P}}(I)$ to derive negative information.

Definition

Operator $\mathbf{U}_{\mathcal{P}}(I) = \{\mathtt{not}.a \mid a \in GUS_{\mathcal{P}}(I)\}$

Unfounded Operator

Theorem

For any program $\mathcal{P}$ and partial interpretation I, the greatest unfounded set $GUS_{\mathcal{P}}(I)$ (which is a superset of all unfounded sets) exists and is unique.

Idea: Use $GUS_{\mathcal{P}}(I)$ to derive negative information.

Definition

Operator $\mathbf{U}_{\mathcal{P}}(I) = \{\mathtt{not}.a \mid a \in GUS_{\mathcal{P}}(I)\}$

Well-Founded Operator

First generalize $\mathbf{T}_{\mathcal{P}}(I)$ for partial interpretations:

Definition

$\mathbf{T}_{\mathcal{P}}(I) := \{h \mid r \in Ground(\mathcal{P}), B(r) \subseteq I, h \in H(r)\}$

Define the well-founded operator $\mathbf{W}_{\mathcal{P}}(I)$ as a combination of $\mathbf{T}_{\mathcal{P}}(I)$ and $\mathbf{U}_{\mathcal{P}}(I)$.

Definition

$\mathbf{W}_{\mathcal{P}}(I) = \mathbf{T}_{\mathcal{P}}(I) \cup \mathbf{U}_{\mathcal{P}}(I)$

Well-Founded Operator

First generalize $\mathbf{T}_{\mathcal{P}}(I)$ for partial interpretations:

Definition

$\mathbf{T}_{\mathcal{P}}(I) := \{h \mid r \in Ground(\mathcal{P}), B(r) \subseteq I, h \in H(r)\}$

Define the well-founded operator $\mathbf{W}_{\mathcal{P}}(I)$ as a combination of $\mathbf{T}_{\mathcal{P}}(I)$ and $\mathbf{U}_{\mathcal{P}}(I)$.

Definition

$\mathbf{W}_{\mathcal{P}}(I) = \mathbf{T}_{\mathcal{P}}(I) \cup \mathbf{U}_{\mathcal{P}}(I)$

Well-Founded Model

Allen Van Gelder

Kenneth Ross

John Schlipf

Well-Founded Model

Theorem

$\mathbf{W}_{\mathcal{P}}$ is monotone and allows for a least fixpoint.

Definition

The least fixpoint $\mathbf{W}_{\mathcal{P}}^{\infty}(\emptyset)$ is the Well-Founded Model of a normal program $\mathcal{P}$.

Well-Founded Model

Theorem

$\mathbf{W}_{\mathcal{P}}$ *is monotone and allows for a least fixpoint.*

Definition

The least fixpoint $\mathbf{W}_{\mathcal{P}}^{\infty}(\emptyset)$ is the Well-Founded Model of a normal program $\mathcal{P}$.

Well-Founded Model – Properties

Theorem

Each normal program has a unique Well-Founded Model.

Well-Founded Model – Properties

Definition

A partial interpretation I is total if $I \cup \mathtt{not}.I = \mathbf{HB}(\mathcal{P})$ (each ground atom is true or false).

Theorem

The Well-Founded Model for positive programs is total and corresponds to its Minimal Model.

Theorem

The Well-Founded Model for stratifiable programs is total and corresponds to its Perfect Model.

Well-Founded Model – Properties

Definition

A partial interpretation I is total if $I \cup \texttt{not}.I = \mathbf{HB}(\mathcal{P})$ (each ground atom is true or false).

Theorem

The Well-Founded Model for positive programs is total and corresponds to its Minimal Model.

Theorem

The Well-Founded Model for stratifiable programs is total and corresponds to its Perfect Model.

Well-Founded Model – Example

Example

$$\begin{array}{l} \mathit{person}(\mathit{nicola}). \\ \mathit{alive}(X) \leftarrow \mathit{person}(X). \\ \mathit{male}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{female}(X). \\ \mathit{female}(X) \leftarrow \mathit{person}(X), \mathtt{not}\ \mathit{male}(X). \end{array}$$

The Well-Founded Model is $\{\mathit{person}(\mathit{nicola}), \mathit{alive}(\mathit{nicola})\}$ and it is not total.

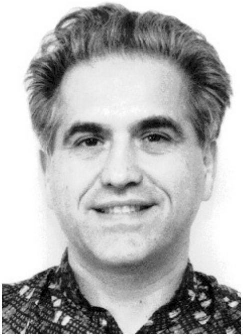
Outline

Stable Models

- No longer a unique model.
- Use total models.
- Stability criterion instead of fixpoint semantics.

Stable Models

Michael Gelfond



Vladimir Lifschitz

Stable Models

Nicole Bidoit

Christine Froidevaux

Gelfond-Lifschitz Reduct

Definition

The Gelfond-Lifschitz reduct of a program $\mathcal{P}^I$ is defined as follows, starting from $Ground(\mathcal{P})$:

1. Delete rules r, for which $B^-(r) \cap I \neq \emptyset$.
2. Delete the negative bodies of the remaining rules.

Gelfond-Lifschitz Reduct

Example

$$\mathcal{P} = \{\ \mathit{male}(g) \leftarrow \mathtt{not}\ \mathit{female}(g).\ \ \mathit{female}(g) \leftarrow \mathtt{not}\ \mathit{male}(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{\mathit{male}(g).\ \mathit{female}(g).\}$
$I_2 = \{\mathit{male}(g)\}, \mathcal{P}^{I_2} = \{\mathit{male}(g).\}$
$I_3 = \{\mathit{female}(g)\}, \mathcal{P}^{I_3} = \{\mathit{female}(g).\}$
$I_4 = \{\mathit{male}(g), \mathit{female}(g)\}, \mathcal{P}^{I_4} = \emptyset$

Gelfond-Lifschitz Reduct

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).\quad female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{male(g).\ female(g).\}$
$I_2 = \{male(g)\}$, $\mathcal{P}^{I_2} = \{male(g).\}$
$I_3 = \{female(g)\}$, $\mathcal{P}^{I_3} = \{female(g).\}$
$I_4 = \{male(g), female(g)\}$, $\mathcal{P}^{I_4} = \emptyset$

Gelfond-Lifschitz Reduct

Example

$$\mathcal{P} = \{ \; male(g) \leftarrow \texttt{not} \; female(g). \quad female(g) \leftarrow \texttt{not} \; male(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{male(g).\ female(g).\}$

$I_2 = \{male(g)\}, \mathcal{P}^{I_2} = \{male(g).\}$

$I_3 = \{female(g)\}, \mathcal{P}^{I_3} = \{female(g).\}$

$I_4 = \{male(g), female(g)\}, \mathcal{P}^{I_4} = \emptyset$

Gelfond-Lifschitz Reduct

Example

$$\mathcal{P} = \{\ \mathit{male}(g) \leftarrow \texttt{not}\ \mathit{female}(g).\quad \mathit{female}(g) \leftarrow \texttt{not}\ \mathit{male}(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{\mathit{male}(g).\ \mathit{female}(g).\}$
$I_2 = \{\mathit{male}(g)\}, \mathcal{P}^{I_2} = \{\mathit{male}(g).\}$
$I_3 = \{\mathit{female}(g)\}, \mathcal{P}^{I_3} = \{\mathit{female}(g).\}$
$I_4 = \{\mathit{male}(g), \mathit{female}(g)\}, \mathcal{P}^{I_4} = \emptyset$

Gelfond-Lifschitz Reduct

Example

$$\mathcal{P} = \{\ \mathit{male}(g) \leftarrow \mathtt{not}\ \mathit{female}(g). \quad \mathit{female}(g) \leftarrow \mathtt{not}\ \mathit{male}(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{\mathit{male}(g).\ \mathit{female}(g).\}$
$I_2 = \{\mathit{male}(g)\}, \mathcal{P}^{I_2} = \{\mathit{male}(g).\}$
$I_3 = \{\mathit{female}(g)\}, \mathcal{P}^{I_3} = \{\mathit{female}(g).\}$
$I_4 = \{\mathit{male}(g), \mathit{female}(g)\}, \mathcal{P}^{I_4} = \emptyset$

Gelfond-Lifschitz Reduct

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).\quad female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{male(g).\ female(g).\}$
$I_2 = \{male(g)\}, \mathcal{P}^{I_2} = \{male(g).\}$
$I_3 = \{female(g)\}, \mathcal{P}^{I_3} = \{female(g).\}$
$I_4 = \{male(g), female(g)\}, \mathcal{P}^{I_4} = \emptyset$

Gelfond-Lifschitz Reduct

Example

$$\mathcal{P} = \{\ \mathit{male}(g) \leftarrow \texttt{not}\ \mathit{female}(g).$$
$$\mathit{female}(g) \leftarrow \texttt{not}\ \mathit{male}(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{\mathit{male}(g).\ \mathit{female}(g).\}$
$I_2 = \{\mathit{male}(g)\}, \mathcal{P}^{I_2} = \{\mathit{male}(g).\}$
$I_3 = \{\mathit{female}(g)\}, \mathcal{P}^{I_3} = \{\mathit{female}(g).\}$
$I_4 = \{\mathit{male}(g), \mathit{female}(g)\}, \mathcal{P}^{I_4} = \emptyset$

Gelfond-Lifschitz Reduct

Example

$$\mathcal{P} = \{\ \mathit{male}(g) \leftarrow \texttt{not}\ \mathit{female}(g). \\ \mathit{female}(g) \leftarrow \texttt{not}\ \mathit{male}(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{\mathit{male}(g).\ \mathit{female}(g).\}$
$I_2 = \{\mathit{male}(g)\}, \mathcal{P}^{I_2} = \{\mathit{male}(g).\}$
$I_3 = \{\mathit{female}(g)\}, \mathcal{P}^{I_3} = \{\mathit{female}(g).\}$
$I_4 = \{\mathit{male}(g), \mathit{female}(g)\}, \mathcal{P}^{I_4} = \emptyset$

Gelfond-Lifschitz Reduct

Example

$$\mathcal{P} = \{\ \mathit{male}(g) \leftarrow \mathtt{not}\ \mathit{female}(g). \\ \mathit{female}(g) \leftarrow \mathtt{not}\ \mathit{male}(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{\mathit{male}(g).\ \mathit{female}(g).\}$
$I_2 = \{\mathit{male}(g)\}, \mathcal{P}^{I_2} = \{\mathit{male}(g).\}$
$I_3 = \{\mathit{female}(g)\}, \mathcal{P}^{I_3} = \{\mathit{female}(g).\}$
$I_4 = \{\mathit{male}(g), \mathit{female}(g)\}, \mathcal{P}^{I_4} = \emptyset$

Stable Models

Fact

Gelfond-Lifschitz reducts are always positive, and have a unique Minimal Model.

Definition

A total interpretation M is a Stable Model of $\mathcal{P}$, if $M = MM(\mathcal{P}^M)$.

Stable Models – Example

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).$$
$$female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{male(g).\ female(g).\}\ , MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{male(g)\}, \mathcal{P}^{I_2} = \{male(g).\}, MM(\mathcal{P}^{I_2}) = I_2$
I_2 is a stable model.
$I_3 = \{female(g)\}, \mathcal{P}^{I_3} = \{female(g).\}, MM(\mathcal{P}^{I_3}) = I_3$
I_3 is a stable model.
$I_4 = \{male(g).\ female(g)\}, \mathcal{P}^{I_4} = \emptyset, MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).\quad female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{male(g).\ female(g).\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{male(g)\}, \mathcal{P}^{I_2} = \{male(g).\}, MM(\mathcal{P}^{I_2}) = I_2$
I_2 is a stable model.
$I_3 = \{female(g)\}, \mathcal{P}^{I_3} = \{female(g).\}, MM(\mathcal{P}^{I_3}) = I_3$
I_3 is a stable model.
$I_4 = \{male(g).\ female(g)\}, \mathcal{P}^{I_4} = \emptyset, MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).$$
$$female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{male(g).\ female(g).\}\ , MM(\mathcal{P}^{I_1}) \neq I_1$

$I_2 = \{male(g)\}, \mathcal{P}^{I_2} = \{male(g).\}, MM(\mathcal{P}^{I_2}) = I_2$

I_2 is a stable model.

$I_3 = \{female(g)\}, \mathcal{P}^{I_3} = \{female(g).\}, MM(\mathcal{P}^{I_3}) = I_3$

I_3 is a stable model.

$I_4 = \{male(g).\ female(g)\}, \mathcal{P}^{I_4} = \emptyset, MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).$$
$$female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{male(g).\ female(g).\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{male(g)\}, \mathcal{P}^{I_2} = \{male(g).\}, MM(\mathcal{P}^{I_2}) = I_2$
I_2 is a stable model.
$I_3 = \{female(g)\}, \mathcal{P}^{I_3} = \{female(g).\}, MM(\mathcal{P}^{I_3}) = I_3$
I_3 is a stable model.
$I_4 = \{male(g).\ female(g)\}, \mathcal{P}^{I_4} = \emptyset, MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).\\ female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{male(g).\ female(g).\}\ , MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{male(g)\}, \mathcal{P}^{I_2} = \{male(g).\}, MM(\mathcal{P}^{I_2}) = I_2$
I_2 is a stable model.
$I_3 = \{female(g)\}, \mathcal{P}^{I_3} = \{female(g).\}, MM(\mathcal{P}^{I_3}) = I_3$
I_3 is a stable model.
$I_4 = \{male(g).\ female(g)\}, \mathcal{P}^{I_4} = \emptyset, MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).\\ female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{male(g).\ female(g).\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{male(g)\}, \mathcal{P}^{I_2} = \{male(g).\}, MM(\mathcal{P}^{I_2}) = I_2$
I_2 is a stable model.
$I_3 = \{female(g)\}, \mathcal{P}^{I_3} = \{female(g).\}, MM(\mathcal{P}^{I_3}) = I_3$
I_3 is a stable model.
$I_4 = \{male(g).\ female(g)\}, \mathcal{P}^{I_4} = \emptyset, MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).\quad female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{male(g).\ female(g).\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{male(g)\}$, $\mathcal{P}^{I_2} = \{male(g).\}$, $MM(\mathcal{P}^{I_2}) = I_2$
I_2 is a stable model.
$I_3 = \{female(g)\}$, $\mathcal{P}^{I_3} = \{female(g).\}$, $MM(\mathcal{P}^{I_3}) = I_3$
I_3 is a stable model.
$I_4 = \{male(g).\ female(g)\}$, $\mathcal{P}^{I_4} = \emptyset$, $MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).$$
$$female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{male(g).\ female(g).\}\ , MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{male(g)\}, \mathcal{P}^{I_2} = \{male(g).\}, MM(\mathcal{P}^{I_2}) = I_2$
I_2 is a stable model.
$I_3 = \{female(g)\}, \mathcal{P}^{I_3} = \{female(g).\}, MM(\mathcal{P}^{I_3}) = I_3$
I_3 is a stable model.
$I_4 = \{male(g).\ female(g)\}, \mathcal{P}^{I_4} = \emptyset, MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).\\ female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{male(g).\ female(g).\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{male(g)\}, \mathcal{P}^{I_2} = \{male(g).\}, MM(\mathcal{P}^{I_2}) = I_2$
I_2 is a stable model.
$I_3 = \{female(g)\}, \mathcal{P}^{I_3} = \{female(g).\}, MM(\mathcal{P}^{I_3}) = I_3$
I_3 is a stable model.
$I_4 = \{male(g).\ female(g)\}, \mathcal{P}^{I_4} = \emptyset, MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).$$
$$female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{male(g).\ female(g).\}\ , MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{male(g)\}, \mathcal{P}^{I_2} = \{male(g).\}, MM(\mathcal{P}^{I_2}) = I_2$
I_2 is a stable model.
$I_3 = \{female(g)\}, \mathcal{P}^{I_3} = \{female(g).\}, MM(\mathcal{P}^{I_3}) = I_3$
I_3 is a stable model.
$I_4 = \{male(g).\ female(g)\}, \mathcal{P}^{I_4} = \emptyset, MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example

$$\mathcal{P} = \{\ male(g) \leftarrow \texttt{not}\ female(g).\quad female(g) \leftarrow \texttt{not}\ male(g).\}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{male(g).\ female(g).\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{male(g)\}$, $\mathcal{P}^{I_2} = \{male(g).\}$, $MM(\mathcal{P}^{I_2}) = I_2$
I_2 is a stable model.
$I_3 = \{female(g)\}$, $\mathcal{P}^{I_3} = \{female(g).\}$, $MM(\mathcal{P}^{I_3}) = I_3$
I_3 is a stable model.
$I_4 = \{male(g).\ female(g)\}$, $\mathcal{P}^{I_4} = \emptyset$, $MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example

$$\mathcal{P} = \{ \mathit{weird} \leftarrow \texttt{not}\ \mathit{weird}. \}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{\mathit{weird}.\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$

$I_2 = \{\mathit{weird}\}$, $\mathcal{P}^{I_2} = \emptyset$, $MM(\mathcal{P}^{I_2}) \neq I_2$

There is no stable model!

Stable Models – Example

Example

$$\mathcal{P} = \{\ \mathit{weird} \leftarrow \mathtt{not}\ \mathit{weird}.\}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{\mathit{weird}.\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$

$I_2 = \{\mathit{weird}\}$, $\mathcal{P}^{I_2} = \emptyset$, $MM(\mathcal{P}^{I_2}) \neq I_2$

There is no stable model!

Stable Models – Example

Example

$$\mathcal{P} = \{\ weird \leftarrow \mathtt{not}\ weird.\}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{weird.\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$

$I_2 = \{weird\}$, $\mathcal{P}^{I_2} = \emptyset$, $MM(\mathcal{P}^{I_2}) \neq I_2$

There is no stable model!

Stable Models – Example

Example

$$\mathcal{P} = \{\ weird \leftarrow \mathtt{not}\ weird.\}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{weird.\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$

$I_2 = \{weird\}$, $\mathcal{P}^{I_2} = \emptyset$, $MM(\mathcal{P}^{I_2}) \neq I_2$

There is no stable model!

Stable Models – Example

Example

$$\mathcal{P} = \{\ \mathit{weird} \leftarrow \mathtt{not}\ \mathit{weird}.\}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{\mathit{weird}.\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$

$I_2 = \{\mathit{weird}\}$, $\mathcal{P}^{I_2} = \emptyset$, $MM(\mathcal{P}^{I_2}) \neq I_2$

There is no stable model!

Stable Models – Example

Example

$$\mathcal{P} = \{\ \mathit{weird} \leftarrow \mathtt{not}\ \mathit{weird}.\}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{\mathit{weird}.\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$

$I_2 = \{\mathit{weird}\}$, $\mathcal{P}^{I_2} = \emptyset$, $MM(\mathcal{P}^{I_2}) \neq I_2$

There is no stable model!

Stable Models – Example

Example

$$\mathcal{P} = \{\, weird \leftarrow \texttt{not}\ weird.\}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{weird.\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{weird\}$, $\mathcal{P}^{I_2} = \emptyset$, $MM(\mathcal{P}^{I_2}) \neq I_2$
There is no stable model!

Stable Models – Example

Example

$$\mathcal{P} = \{\ weird \leftarrow \texttt{not}\ weird.\}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{weird.\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$
$I_2 = \{weird\}$, $\mathcal{P}^{I_2} = \emptyset$, $MM(\mathcal{P}^{I_2}) \neq I_2$
There is no stable model!

Stable Models

Theorem

For positive programs there is exactly one Stable Model, which is equal to the Minimal Model.

Theorem

For stratifiable programs there is exactly one Stable Model, which is equal to the Perfect Model.

Stable Models

Theorem

For positive programs there is exactly one Stable Model, which is equal to the Minimal Model.

Theorem

For stratifiable programs there is exactly one Stable Model, which is equal to the Perfect Model.

Stable Models

Theorem

If the Well-Founded Model of a program is total, then the program has a corresponding unique Stable Model.

Theorem

The positive part of the Well-Founded Model of a program is contained in each Stable Model of the program.

Stable Models

Theorem

If the Well-Founded Model of a program is total, then the program has a corresponding unique Stable Model.

Theorem

The positive part of the Well-Founded Model of a program is contained in each Stable Model of the program.

Stable Models – Consequences

Definition (Brave/Credulous Reasoning)

$\mathcal{P} \models_b l$ iff l is true in some Stable Model of $\mathcal{P}$.

Definition (Cautious/Skeptical Reasoning)

$\mathcal{P} \models_c l$ iff l is true in all Stable Models of $\mathcal{P}$.

Note: If $\mathcal{P}$ admits no Stable Model, then all literals are cautious/skeptical consequences!

Stable Models – Consequences

Definition (Brave/Credulous Reasoning)

$\mathcal{P} \models_b l$ iff l is true in some Stable Model of $\mathcal{P}$.

Definition (Cautious/Skeptical Reasoning)

$\mathcal{P} \models_c l$ iff l is true in all Stable Models of $\mathcal{P}$.

Note: If $\mathcal{P}$ admits no Stable Model, then all literals are cautious/skeptical consequences!

Stable Models – Example

Example (Two-Colorability)

Given a graph, can each vertex be assigned one of two colors, such that adjacent vertices do not have the same color?

$$
\begin{array}{l}
vertex(V) \leftarrow arc(V, Y).\ vertex(V) \leftarrow arc(X, V).\\
color(V, white) \leftarrow vertex(V), \mathtt{not}\ color(V, black).\\
color(V, black) \leftarrow vertex(V), \mathtt{not}\ color(V, white).\\
bad \leftarrow color(V1, F), color(V2, F),\\
\qquad arc(V1, V2), \mathtt{not}\ bad.
\end{array}
$$

Answer Set Programming

For several people Answer Set Programming is equal to Datalog with negation under the stable model semantics!

For me and many others it is more, though.

Answer Set Programming

For several people Answer Set Programming is equal to Datalog with negation under the stable model semantics!

For me and many others it is more, though.

Part II

Answer Set Programming

Outline

Answer Set Programming

A disjunctive rule is

$$h_1 \mid \dots \mid h_k \leftarrow b_1, \dots, b_m, \texttt{not } b_{m+1}, \dots, \texttt{not } b_n.$$
$$1 \leqslant k; 1 \leqslant m \leqslant n$$

Let
$H(r) = \{h_1, \dots, h_k\}$
everything else as before

Disjunctive Programs – Semantics

- Most concepts do not change.
- Satisfaction of a rule r with respect to M:
  If $B^+(r) \subseteq M$ and $M \cap B^-(r) = \emptyset$, then $H(r) \subseteq M$
- Reduct?

Gelfond-Lifschitz Reduct

Definition

The Gelfond-Lifschitz reduct of a program $\mathcal{P}^I$ is defined as follows, starting from $Ground(\mathcal{P})$:

1. Delete rules r, for which $B^-(r) \cap I \neq \emptyset$.
2. Delete the negative bodies of the remaining rules.

Same as without disjunction!

Stable Models

Fact

Gelfond-Lifschitz reducts are always positive, and have multiple Minimal Models.

Definition

A total interpretation M is a Stable Model of $\mathcal{P}$, if $M \in MM(\mathcal{P}^M)$.

Stable Models – Example

Example (Two-Colorability)

Given a graph, can each vertex be assigned one of two colors, such that adjacent vertices do not have the same color?

$$
\begin{array}{l}
\mathit{vertex}(V) \leftarrow \mathit{arc}(V, Y). \ \mathit{vertex}(V) \leftarrow \mathit{arc}(X, V). \\
\mathit{color}(V, \mathit{white}) \mid \mathit{color}(V, \mathit{black}) \leftarrow \mathit{vertex}(V). \\
\mathit{bad} \leftarrow \mathit{color}(V1, F), \mathit{color}(V2, F), \\
\qquad \mathit{arc}(V1, V2), \mathtt{not}\ \mathit{bad}.
\end{array}
$$

Note: No stable model will contain both $\mathit{color}(v, \mathit{white})$ and $\mathit{color}(v, \mathit{black})$ for any vertex v due to minimality!

Stable Models – Example

Example (Two-Colorability)

Given a graph, can each vertex be assigned one of two colors, such that adjacent vertices do not have the same color?

$$\begin{array}{l} \mathit{vertex}(V) \leftarrow \mathit{arc}(V,Y).\ \mathit{vertex}(V) \leftarrow \mathit{arc}(X,V). \\ \mathit{color}(V,\mathit{white}) \mid \mathit{color}(V,\mathit{black}) \leftarrow \mathit{vertex}(V). \\ \mathit{bad} \leftarrow \mathit{color}(V1,F), \mathit{color}(V2,F), \\ \qquad \mathit{arc}(V1,V2), \mathtt{not}\ \mathit{bad}. \end{array}$$

Can we always convert disjunctions to negations in this way (shifting)?

Stable Models – Example

Example (Two-Colorability)

Given a graph, can each vertex be assigned one of two colors, such that adjacent vertices do not have the same color?

$$
\begin{aligned}
& vertex(V) \leftarrow arc(V,Y).\ vertex(V) \leftarrow arc(X,V).\\
& color(V,white) \leftarrow vertex(V), \mathtt{not}\ color(V,black).\\
& color(V,black) \leftarrow vertex(V), \mathtt{not}\ color(V,white).\\
& bad \leftarrow color(V1,F), color(V2,F),\\
& \qquad arc(V1,V2), \mathtt{not}\ bad.
\end{aligned}
$$

Can we always convert disjunctions to negations in this way (shifting)?

Stable Models – Example

Example (Shifting)

$$a \mid b.$$
$$a \leftarrow b.$$
$$b \leftarrow a.$$

One answer set: $\{a, b\}$

Stable Models – Example

Example (Shifting)

$$
\begin{aligned}
a &\leftarrow \texttt{not } b.\\
b &\leftarrow \texttt{not } a.\\
a &\leftarrow b.\\
b &\leftarrow a.
\end{aligned}
$$

No answer set! Why?

Stable Models – Example

Example (Shifting)

$$a \mid b.$$
$$a \leftarrow b.$$
$$b \leftarrow a.$$

One answer set: $\{a, b\}$

There is a cycle among the disjunctive atoms!

Head-Cycle Free (HCF) Programs

Definition

P is head-cycle free (HCF) if there is a level mapping $\|.\|_h$ of P such that for every rule $r \in P$:

1. For any $l \in B^+(r)$, and for any $l' \in H(r)$, $\|l\|_h \leqslant \|l'\|_h$
2. For any $l, l' \in H(r)$, $\|l\|_h <> \|l'\|_h$

Theorem

Every head-cycle free program is equivalent to its (non-disjunctive) shifted program.

Head-Cycle Free (HCF) Programs

Example (HCF Program)

$$a \mid b.$$
$$a \leftarrow b.$$

is HCF since:

$\|a\|_h = 2; \quad \|b\|_h = 1$

Example (Non-HCF Program)

$$a \mid b.$$
$$a \leftarrow b.$$
$$b \leftarrow a.$$

No HCF level mapping exists!

Outline

Integrity Constraints

An integrity constraint is

$$\leftarrow b_1, \ldots, b_m, \mathtt{not}\ b_{m+1}, \ldots, \mathtt{not}\ b_n.$$

we view it as a shorthand for

$$bad \leftarrow b_1, \ldots, b_m, \mathtt{not}\ b_{m+1}, \ldots, \mathtt{not}\ b_n, \mathtt{not}\, bad.$$

where *bad* is a reserved predicate.

Outline

True/Strong/Classical Negation

In place of atoms $a(t_1, \dots, t_n)$ one can use also strong literals $\neg a(t_1, \dots, t_n)$.
No answer set should contain both $a(t_1, \dots, t_n)$ and $\neg a(t_1, \dots, t_n)$ of any kind.

Compile that away:

- Replace any $\neg a(t_1, \dots, t_n)$ by $n\_a(t_1, \dots, t_n)$ (n_a a new predicate)
- Add $\leftarrow a(X_1, \dots, X_n), n\_a(X_1, \dots, X_n).$ for each predicate a.

True/Strong/Classical Negation

In place of atoms $a(t_1, \ldots, t_n)$ one can use also strong literals $\neg a(t_1, \ldots, t_n)$.
No answer set should contain both $a(t_1, \ldots, t_n)$ and $\neg a(t_1, \ldots, t_n)$ of any kind.

Compile that away:

- Replace any $\neg a(t_1, \ldots, t_n)$ by $n\_a(t_1, \ldots, t_n)$ (n_a a new predicate)
- Add $\leftarrow a(X_1, \ldots, X_n), n\_a(X_1, \ldots, X_n).$ for each predicate a.

Outline

Weak Constraints

$$\leftsquigarrow b_1, \ldots, b_m, \mathtt{not}\ b_{m+1}, \ldots, \mathtt{not}\ b_n.$$

Constraints that should be satisfied.

- Non-satisfaction can incur a weight
- possibly of a priority level

Produces an ordering of answer sets, identify answer sets.

Weak Constraints

$$\leftsquigarrow b_1, \ldots, b_m, \texttt{not}\ b_{m+1}, \ldots, \texttt{not}\ b_n.[w]$$

Constraints that should be satisfied.

- Non-satisfaction can incur a weight
- possibly of a priority level

Produces an ordering of answer sets, identify answer sets.

Weak Constraints

$$\leftarrow\!\sim b_1, \ldots, b_m, \mathtt{not}\ b_{m+1}, \ldots, \mathtt{not}\ b_n.[w@p]$$

Constraints that should be satisfied.

- Non-satisfaction can incur a weight
- possibly of a priority level

Produces an ordering of answer sets, identify answer sets.

Weak Constraints

$$\leadsto b_1, \ldots, b_m, \mathtt{not}\ b_{m+1}, \ldots, \mathtt{not}\ b_n.[w@p]$$

Constraints that should be satisfied.

- Non-satisfaction can incur a weight
- possibly of a priority level

Produces an ordering of answer sets, identify optimal answer sets.

Weak Constraints

$$\leftsquigarrow b_1, \ldots, b_m, \mathtt{not}\ b_{m+1}, \ldots, \mathtt{not}\ b_n.[w@p]$$

Constraints that should be satisfied.

- Non-satisfaction can incur a weight
- possibly of a priority level

Produces an ordering of answer sets, identify answer sets.

Weak Constraints – Example

Example (Two-Colorability)

Given a graph, can each vertex be assigned one of two colors, such that adjacent vertices do not have the same color, preferring black?

$$\begin{array}{l} vertex(V) \leftarrow arc(V,Y).\ vertex(V) \leftarrow arc(X,V). \\ color(V,white) \mid color(V,black) \leftarrow vertex(V). \\ \leftarrow color(V1,F), color(V2,F), arc(V1,V2). \\ \leftarrow\!\!\sim\ \mathtt{not}\ color(V,black). \end{array}$$

Outline

Combined and Data Complexity

$$a(0).$$
$$a(1).$$
$$b(X_1, \ldots, X_n) \leftarrow a(X_1), \ldots, a(X_n).$$

Consider data complexity, or equivalently variable-free programs!

Combined and Data Complexity

$$
\begin{array}{l}
a(0).\\
a(1).\\
b(X_1,\ldots,X_n) \leftarrow a(X_1),\ldots,a(X_n).
\end{array}
$$

Consider data complexity, or equivalently variable-free programs!

Intuitive explanation

Three main sources of complexity:

- the exponential number of answer set “candidates”
- checking whether a candidate M is an answer set of P (minimality of M can be disproved by exponentially many subsets of M)
- checking optimality of the answer set w.r.t. the violation of the weak constraints

Complexity – Answer Set Checking

	$\{\}$	$\{w\}$	$\{\mathtt{not}_s\}$	$\{\mathtt{not}_s, w\}$	$\{\mathtt{not}\}$	$\{\mathtt{not}, w\}$
$\{\}$	P	P	P	P	P	co-NP
$\{\,\|_{\mathtt{h}}\,\}$	P	co-NP	P	co-NP	P	co-NP
$\{\,\|\,\}$	co-NP	Π_2^P	co-NP	Π_2^P	co-NP	Π_2^P

Complexity – Brave Reasoning

	$\{\}$	$\{w\}$	$\{\mathtt{not}_s\}$	$\{\mathtt{not}_s, w\}$	$\{\mathtt{not}\}$	$\{\mathtt{not}, w\}$
$\{\}$	P	P	P	P	NP	Δ_2^P
$\{\mid_{\mathtt{h}}\}$	NP	Δ_2^P	NP	Δ_2^P	NP	Δ_2^P
$\{\mid\}$	Σ_2^P	Δ_3^P	Σ_2^P	Δ_3^P	Σ_2^P	Δ_3^P

Complexity – Cautious Reasoning

	$\{\}$	$\{w\}$	$\{\mathtt{not}_s\}$	$\{\mathtt{not}_s, w\}$	$\{\mathtt{not}\}$	$\{\mathtt{not}, w\}$
$\{\}$	P	P	P	P	co-NP	Δ_2^P
$\{\,\vert_{\mathtt{h}}\,\}$	co-NP	Δ_2^P	co-NP	Δ_2^P	co-NP	Δ_2^P
$\{\,\vert\,\}$	co-NP	Δ_3^P	Π_2^P	Δ_3^P	Π_2^P	Δ_3^P

Outline

Expressivity?

What do we mean by expressivity or capturing?

Given a problem P in complexity class X, can we find an ASP program Π_P such that for any input I encoded as facts Π_I, the answer sets of $\Pi_I \cup \Pi_P$ are in a 1-1 correspondence to the solutions of P on input I?

Except for classes without negation, the fragments of ASP capture the classes for which they are complete.

Expressivity?

What do we mean by expressivity or capturing?
Given a problem P in complexity class X, can we find an ASP program Π_P such that for any input I encoded as facts Π_I, the answer sets of $\Pi_I \cup \Pi_P$ are in a 1-1 correspondence to the solutions of P on input I?
Except for classes without negation, the fragments of ASP capture the classes for which they are complete.

Expressivity?

What do we mean by expressivity or capturing?

Given a problem P in complexity class X, can we find an ASP program Π_P such that for any input I encoded as facts Π_I, the answer sets of $\Pi_I \cup \Pi_P$ are in a 1-1 correspondence to the solutions of P on input I?

Except for classes without negation, the fragments of ASP capture the classes for which they are complete.

Outline

Aggregate Atom

$$L_g <_1 f\{S\} <_2 U_g$$

$$5 < \#count\{EmpId : emp(EmpId, Male, Skill, Salary)\} \leqslant 10$$

The atom is true if the number of male employees is greater than 5 and does not exceed 10.

Aggregate Example

Example (Team Building)

% An employee is either included in the team or not
$inTeam(I) \mid outTeam(I) \leftarrow emp(I, Sx, Sk, Sa).$

% The team consists of a certain number of employees
$\leftarrow nEmp(N), \mathtt{not}\ \#count\{I : inTeam(I)\} = N.$

% At least a given number of different skills must be present in the team
$\leftarrow nSkill(M), \mathtt{not}\ \#count\{Sk : emp(I, Sx, Sk, Sa), inTeam(I)\} \leqslant M.$

% The sum of the salaries of the employees working in the team must not exceed the given budget
$\leftarrow budget(B), \mathtt{not}\ \#sum\{Sa, I : emp(I, Sx, Sk, Sa), inTeam(I)\} \leqslant B.$

% The salary of each individual employee is within a specified limit
$\leftarrow maxSal(M), \mathtt{not}\ \#max\{Sa : emp(I, Sx, Sk, Sa), inTeam(I)\} \leqslant M.$

Recursive Aggregates

Example

$$a(1) \leftarrow \#\mathtt{count}\{X : a(X)\} \geqslant 1.$$

intuitively equivalent to

$$a(1) \leftarrow a(1).$$

One expected answer set: $\emptyset$
Treating aggregates like negative literals yields two answer sets: $\emptyset$ and $\{a(1)\}$!

Recursive Aggregates

Example

$$a(1) \leftarrow \#\mathtt{count}\{X : a(X)\} \geqslant 1.$$

intuitively equivalent to

$$a(1) \leftarrow a(1).$$

One expected answer set: $\emptyset$
Treating aggregates like negative literals yields two answer sets: $\emptyset$ and $\{a(1)\}$!

Recursive Aggregates

Example

$$a(1) \leftarrow \#\mathtt{count}\{X : a(X)\} < 1.$$

intuitively equivalent to

$$a(1) \leftarrow \mathtt{not}\ a(1).$$

Expected answer sets: none

Treating aggregates like positive literals yields one answer set: $\{a(1)\}$!

Recursive Aggregates

Example

$$a(1) \leftarrow \#\mathtt{count}\{X : a(X)\} < 1.$$

intuitively equivalent to

$$a(1) \leftarrow \mathtt{not}\ a(1).$$

Expected answer sets: none
Treating aggregates like positive literals yields one answer set: $\{a(1)\}$!

Aggregate Semantics

The **FLP Reduct** (F., Leone, Pfeifer) of a ground program P w.r.t. a set X is the positive ground program P^X obtained from P by:

- deleting all rules with a false literal in the body (w.r.t. X);

Answer Set: An *answer set* of a program P is a set $X \subseteq BP$ such that X is a minimal model of P^X.

Equivalent to Gelfond-Lifschitz reduct on aggregate-free programs

Can be used for any "generalized atoms": *HEX atoms*, *DL atoms* etc.

Aggregate Semantics

The **FLP Reduct** (F., Leone, Pfeifer) of a ground program P w.r.t. a set X is the positive ground program P^X obtained from P by:

- deleting all rules with a false literal in the body (w.r.t. X);

Answer Set: An *answer set* of a program P is a set $X \subseteq BP$ such that X is a minimal model of P^X.

Equivalent to Gelfond-Lifschitz reduct on aggregate-free programs

Can be used for any “generalized atoms”: *HEX atoms*, *DL atoms* etc.

Aggregate Semantics

The **FLP Reduct** (F., Leone, Pfeifer) of a ground program P w.r.t. a set X is the positive ground program P^X obtained from P by:

- deleting all rules with a false literal in the body (w.r.t. X);

Answer Set: An *answer set* of a program P is a set $X \subseteq BP$ such that X is a minimal model of P^X.

Equivalent to Gelfond-Lifschitz reduct on aggregate-free programs
Can be used for any “generalized atoms”: *HEX atoms*, *DL atoms* etc.

DL Atoms

$$DL[S_1 \uplus p_1, S_2 \cupdot p_2, S_3 \cap\!\!\!\!- p_3, \ldots; Q](t_1, \ldots, t_n)$$

Evaluate DL query Q over a given ontology, adding positive/negative assertions to concepts/roles:

$S_1, \ldots$: concepts/roles
$p_1, \ldots$: unary/binary predicates

Can be treated like "fancy" aggregates.
Satisfied in I iff

$$(\mathcal{T}, \mathcal{A} \cup \{S_1(\overline{u}) \mid p_1(\overline{u}) \in I\} \cup \{\neg S_2(\overline{u}) \mid p_2(\overline{u}) \in I\}$$
$$\cup \{\neg S_3(\overline{u}) \mid p_3(\overline{u}) \notin I\} \cup \ldots \models Q(t_1, \ldots, t_n)$$

DL Atoms

$$DL[S_1 \uplus p_1, S_2 \uminus p_2, S_3 \cap\!\!\!\!- \, p_3, \ldots; Q](t_1, \ldots, t_n)$$

Evaluate DL query Q over a given ontology, adding positive/negative assertions to concepts/roles:

$S_1, \ldots$: concepts/roles
$p_1, \ldots$: unary/binary predicates

Can be treated like “fancy” aggregates.
Satisfied in I iff

$$(\mathcal{T}, \mathcal{A} \cup \{S_1(\overline{u}) \mid p_1(\overline{u}) \in I\} \cup \{\neg S_2(\overline{u}) \mid p_2(\overline{u}) \in I\} \cup \{\neg S_3(\overline{u}) \mid p_3(\overline{u}) \notin I\} \cup \ldots \models Q(t_1, \ldots, t_n)$$

DL Atoms

$$DL[S_1 \uplus p_1, S_2 \uminus p_2, S_3 \cap\!\!\!\!- p_3, \ldots ; Q](t_1, \ldots, t_n)$$

Evaluate DL query Q over a given ontology, adding positive/negative assertions to concepts/roles:

$S_1, \ldots$: concepts/roles
$p_1, \ldots$: unary/binary predicates

Can be treated like “fancy” aggregates.
Satisfied in I iff

$$(\mathcal{T}, \mathcal{A} \cup \{S_1(\overline{u}) \mid p_1(\overline{u}) \in I\} \cup \{\neg S_2(\overline{u}) \mid p_2(\overline{u}) \in I\} \cup \{\neg S_3(\overline{u}) \mid p_3(\overline{u}) \notin I\} \cup \ldots \models Q(t_1, \ldots, t_n)$$

More Semantics

Several more semantics have been proposed for generalized programs:

- Pelov; Son and Pontelli
- Eiter et al. (strong and weak semantics)
- Shen and colleagues
- ...

All reasonable ones coincide on standard programs and programs with stratified general atoms.

Monotonicity

General atoms can be

- Monotonic
  truth in I implies truth in all $J \supseteq I$
- Antimonotonic
  truth in I implies truth in all $J \subseteq I$
- Nonmonotonic
  neither monotonic nor antimonotonic
- Convex
  truth in I and $J \supseteq I$ implies truth in all K s.t. $I \subseteq K \subseteq J$

All reasonable semantics coincide on programs without nonmonotonic general atoms.

Probably also on convex general atoms.

Monotonicity

General atoms can be

- Monotonic
  truth in I implies truth in all $J \supseteq I$
- Antimonotonic
  truth in I implies truth in all $J \subseteq I$
- Nonmonotonic
  neither monotonic nor antimonotonic
- Convex
  truth in I and $J \supseteq I$ implies truth in all K s.t. $I \subseteq K \subseteq J$

All reasonable semantics coincide on programs without nonmonotonic general atoms.
Probably also on convex general atoms.

Outline

Choice Rules

$$\{h_1, \ldots, h_k\} \leftarrow b_1, \ldots, b_m, \mathtt{not}\ b_{m+1}, \ldots, \mathtt{not}\ b_n.$$

If the body is true, any subset of $\{h_1, \ldots, h_k\}$ must be true.

$$l\{h_1, \ldots, h_k\}u \leftarrow b_1, \ldots, b_m, \mathtt{not}\ b_{m+1}, \ldots, \mathtt{not}\ b_n.$$

If the body is true, between l and u atoms of $\{h_1, \ldots, h_k\}$ must be true (inclusively).

Choice Rules

$$\{h_1, \ldots, h_k\} \leftarrow b_1, \ldots, b_m, \mathtt{not}\ b_{m+1}, \ldots, \mathtt{not}\ b_n.$$

If the body is true, any subset of $\{h_1, \ldots, h_k\}$ must be true.

$$l\{h_1, \ldots, h_k\}u \leftarrow b_1, \ldots, b_m, \mathtt{not}\ b_{m+1}, \ldots, \mathtt{not}\ b_n.$$

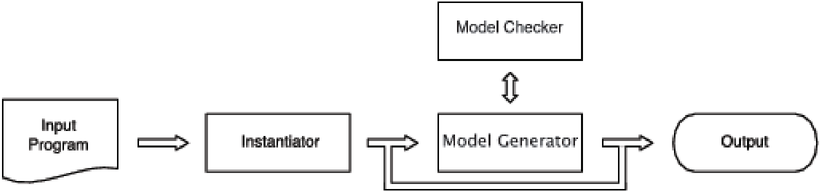
If the body is true, between l and u atoms of $\{h_1, \ldots, h_k\}$ must be true (inclusively).

Outline

ASP Computation

Computationally expensive
Traditionally a two-step process:

1. **Instantiation (grounder)**
   Variable elimination
2. **Propositional search (solver)**
   - Model Generation: generate candidate answer sets
   - Model Checking: verify stability



Outline

Equivalence

$$a \mid b. \qquad\qquad \begin{array}{l} a \leftarrow \texttt{not}\ b. \\ b \leftarrow \texttt{not}\ a. \end{array}$$

Equivalent, both programs have answer sets $\{a\}$ and $\{b\}$.

But the substitution theorem does not hold: the left extended program has answer set $\{a, b\}$, the right one no answer set.

Equivalence

$a \mid b.$	$a \leftarrow \mathtt{not}\ b.$
	$b \leftarrow \mathtt{not}\ a.$
$a \leftarrow b.$	$a \leftarrow b.$
$b \leftarrow a.$	$b \leftarrow a.$

Equivalent, both programs have answer sets $\{a\}$ and $\{b\}$.

But the substitution theorem does not hold: the left extended program has answer set $\{a, b\}$, the right one no answer set.

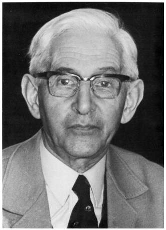
Strong Equivalence

Strong Equivalence: replaceability in any context (substitution theorem holds)

Theorem

P and Q are strongly equivalent iff $P \equiv_{HT} Q$.

HT: Logic of Here and There, Heyting 1930
a.k.a. Gödel Logic G_3



Answer Sets are actually HT models that satisfy an equilibrium condition.

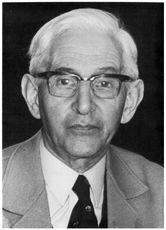
Strong Equivalence

Strong Equivalence: replaceability in any context (substitution theorem holds)

Theorem

P and Q are strongly equivalent iff $P \equiv_{HT} Q$.

HT: Logic of Here and There, Heyting 1930
a.k.a. Gödel Logic G_3



Answer Sets are actually HT models that satisfy an equilibrium condition.

Outline

ASP Systems

- DLV (grounder+solver)
- wasp (solver)
- gringo (grounder)
- clasp (solver)
- cmodels (solver)
- lparse (grounder)
- smodels (solver)
- IDP (grounder+solver)
- ...

Techniques

- Deductive database techniques
- Magic Sets
- Techniques from SAT
- Techniques from CSP

Support

- Development environments: e.g. ASPIDE
- Application embedding: e.g. JASP
- Debuggers
- Visualizers

Outline

ASP Competition

- Biannual
- `https://www.mat.unical.it/aspcomp2013/`
- System Track
- Model & Solve Track

The System Track gave rise to the first serious language standard.

https://www.mat.unical.it/aspcomp2013/ASPStandardization

Topics Not Covered

Of course incomplete...

- ASP with function symbols
- ASP with existential quantification in rule heads
- ASP for arbitrary formulas
- ASP without Unique Name Assumption
- ASP and preferences
- ASP and AI tasks
- ASP applications

Conclusions

- ASP is Datalog with negation, disjunction etc. under the stable model semantics
- For the Web:
  - As a target to rewrite OBDA queries to
  - Loose coupling between ontologies and rules
- Efficient systems
- Development tools available

Try it!

Further Resources

- Nicola Leone and Francesco Ricca's RR 2013 tutorial: `https://www.mat.unical.it/ricca/downloads/rr2013-tutorial.pdf`
- Marin Gebser and Torsten Schaub's IJCAI 2013 tutorial: `http://www.cs.uni-potsdam.de/~torsten/ijcai13tutorial/`